Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado

Elena Campo León

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Director del trabajo: José Tomás Alcalá Nalvaiz

Prólogo

El aprendizaje estadístico supervisado es un conjunto de técnicas para deducir una función a partir de datos de entrenamiento y es una de las herramientas principales de la minería de datos y del aprendizaje automático. Los datos de entrenamiento, ejemplos o instancias consisten en pares de objetos (normalmente vectores) en los que una componente del par son los datos de entrada y el otro, los resultados deseados. El objetivo es crear o estimar una función capaz de predecir el valor deseado correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos. Para ello, tiene que generalizar a partir de los datos presentados anteriormente a las nuevas situaciones no vistas previamente. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación) [WAS].

En este contexto, las **máquinas de vector soporte** (Support Vector Machines, SVMs) son un conjunto de algoritmos de aprendizaje estadístico supervisado pertenecientes a la familia de los clasificadores lineales desarrollados por Vladimir Vapnik y su equipo en los laboratorios $AT\&T$ entorno a 1995.

Sin pérdida de generalidad, suponiendo que tenemos ejemplos de sólo dos categorías una SVM construye un hiperplano en un espacio de dimensionalidad muy alta o incluso infinita. Este hiperplano separa de forma óptima los puntos de una clase de la de otra. En el concepto de "separación óptima" es donde reside la característica fundamental de las SVM, se busca el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Por eso también a veces se les conoce a las SVM como clasificadores de margen máximo [WSVM].

Las SVM se desarrollaron inicialmente para resolver problemas de clasificación. Sin embargo, se han reformulado de múltiples formas a lo largo de los años, sus variantes mas conocidas son: SVM para regresión, SVM para resolución de ecuaciones integrales, SVM para estimar el soporte de una densidad, SVMs que usan diferentes costes de margen blando y parámetros. También se han ensayado otras formulaciones del problema dual (ver [BL]). Las técnicas de óptimización cuadrática son fundamentales a la hora de resolver estos problemas.

Actualmente, las SVM tienen utilidad en una extensa variedad de áreas. Algunas de sus aplicaciones mas importantes son el reconocimiento de caracteres, detección de intrusos, reconocimiento del habla o la bioinformática.

Summary

The Support Vector Machine (SVM) algorithm is probably the most widely used kernel learning algorithm. It achieves relatively robust pattern recognition performance using well established concepts in optimization theory. The pattern recognition has a wide range of applications including optical character recognition, intrusion detection, speech recognition, and bioinformatics.

Statistical learning is consolidated as a branch of Statistics around 1995 when Vladimir Vapnik published his renowned book "The Nature of Statistical Learning Theory". Supervised statistical learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. Each example or instance is a pair consisting of an input object, typically a vector $\mathbf{x}$, and a desired output value y. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations. The parallel task in human and animal psychology is often referred to as concept learning.

This work shows an introduction to the extensive field of support vector machines and its utility as classifiers. In addition, we will see a practical application in medicine to a clinical case of patients with breast cancer. The model is build using learnig algorithms. In this way if a new patient is received, our classifier will be able to predict if she is suffering a malignant or benign tumor.

In chapter 1, we present the earliest pattern regognition systems, binary classifiers. For each pattern $\mathbf{x} \in \mathbb{R}^d$ is given a class $y \in \{-1, 1\}$. These examples have the property of being perfectly and linearly separable, so they will be classified without error by a linear discriminant function $D(\mathbf{x}) = (w_1 x_1 + \ldots + w_d x_d) + b$, where $\mathbf{w} \in \mathbb{R}^d$ and b is a real coefficient. When a training set is linearly separable there usually is an infinity of separating hyperplanes. Vapnik and other authors propose to choose the separating hyperplane that maximizes the margin, that is to say the hyperplane that leaves as much room as possible between the hyperplane and the closest example. Formally, this situation is represented as an optimization problem in which we minimize the norm of the hyperplane's directional vector $\mathbf{w}$ under some constraints. However this state of perfect separability does not happen in real life. Usually all examples may not be correctly classified. In that case we have two solutions, the choice of each one depend on the degree of separability of the examples.

In the first situation, when the problem is very noisy is usually to address it by allowing some examples to violate the margin constraints in the optimization problem. These potential violations are represented using positive slack variables ξ_i, $i = 1, \ldots, n$. An additional parameter called cost C controls the compromise between large margins and small margin violations.

The formulation of the optimization problem is the following:

$$\begin{aligned} \text{mín} \qquad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.a \qquad & y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) + \xi_i - 1 \geq 0, \\ & \xi_i \geq 0, \qquad i = 1,\ldots,n. \end{aligned} \tag{1}$$

In the second case the set of examples cannot be separated by a linear function, so we use the so-called kernel functions, which represent a scalar product. Kernel function induces a high dimensional Hilbert space, called features space. Each pattern $\mathbf{x}$ is transformed into a feature vector $\Phi(\mathbf{x}) \in \mathbb{R}^m$, so the linear discriminant function will be $D'(\mathbf{x}) = (w_1\phi_1(\mathbf{x} + \cdots + w_m\phi_m(\mathbf{x})) + b$. In the original space this hyperplane will be transformed into a non linear decision boundary. The Reproducing Kernel Hilbert Spaces theory states that, although the corresponding feature space has infinite dimension, if we know the kernel function, all computations can be performed without ever computing a feature vector $\Phi(\mathbf{x})$.

In both cases the optimization problem (primal) is transformed to its dual, so we can solve it by quadratic optimization techniques. Dual problem is computationally easier because the number of variables to optimize in the primal problem is directly proportional to the dimensionality of the examples while in the dual problem it is directly proportional with the size of the sample, so the constraints are much simple.

In chapter 2, we present two main extensions of SVM: multiclass classifiers and SVR for regression. Support vector machine originally separates the binary classes with a maximized margin criterion. However, real-world problems often require the discrimination for $k \geq 2$ categories. The most used multiclass solving technique is SimMSVM. This method designs a unique objective function that trains simultaneously $\frac{k(k-1)}{2}$ binary classifiers and through a voting strategy is assigned to each example its label. We obtain a quadratic optimization problem with a higher misclassification rate than other methods, in exchange of, reducing the number of variables to the number of examples. In conclusion SimMSVM approach can greatly speed up the training process, while maintaining a competitive classification accuracy.

The other extension, SVR, is the adaptation of the original SVM problem to solving a regression problem. In this case we have $y_i \in \mathbb{R}$, for $i = 1\ldots,n$, instead of a class label. We define a loss function L_ϵ that controls the error between the predicted and actual value. This function only takes non-zero values when the separation distance between the regression hyperplane and the example is greater than the parameter ϵ. In the same way to the classification problem the slack variables associated to each example are used to allow certain prediction error.

In chapter 3 we make a brief summary about currently packages containing SVM related software and specially for the statistical software package **R**, the e1071 library among others. Using a popular dataset of patients with breast cancer, we optimize several models for different kernel functions and we obtain the corresponding ROC curves to analyze the quality of each one. In all three models, we have excellent results about the accuracity of the true diagnosis with respect to the type of tumor (malignant or benign) for a new patient.

Índice general

Capítulo 1

SVM para clasificación binaria

1.1. Ejemplos separables linealmente

Consideramos un conjunto de elementos representados por el par $(\mathbf{x_i}, y_i)$ que llamaremos **ejemplos**, cada uno con una serie de características que se almacenan en un vector $\mathbf{x_i}$. Además tienen asignada una categoría que es la segunda componente de $(\mathbf{x_i}, y_i)$. Observemos la figura 1.1 (a), tenemos dos clases de ejemplos, si podemos separar todos los ejemplos a través de un hiperplano, de forma que, cada uno este en el semiplano de la clase correcta, entonces diremos que el conjunto de ejemplos **es separable**.

Dado un conjunto separable de ejemplos, $S = \{(\mathbf{x_1}; y_1), \cdots, (\mathbf{x_n}; y_n)\}$, donde $\mathbf{x_i} \in \mathbb{R}^d$ e $y_i \in \{-1, 1\}$, se define un **hiperplano de separación** como una función lineal que es capaz de separar dicho conjunto sin error:

$$D(\mathbf{x_i}) = (w_1 x_1 + \ldots + w_d x_d) + b = \langle \mathbf{w}, \mathbf{x_i} \rangle + b, \tag{1.1}$$

donde $\mathbf{w} \in \mathbb{R}^d$ y b es un coeficiente real. Las restricciones que debe satisfacer un hiperplano de separación para todo ejemplo del conjunto de entrenamientos son:

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x_i} \rangle + b &\geqslant 0, \qquad \text{si } \; y_i = +1, \\ \langle \mathbf{w}, \mathbf{x_i} \rangle + b &\leqslant 0, \qquad \text{si } \; y_i = -1. \end{aligned}$$

Equivalentemente,

$$y_i D(\mathbf{x_i}) \geqslant 0, \qquad i = 1, \ldots, n. \tag{1.2}$$

Sin embargo existen infinitos hiperplanos (ver figura 1.1 (b)) que satisfacen la restricción anterior, debemos buscar un criterio que nos proporcione una regla de decisión sobre el conjunto de ejemplos, de forma que hallemos el hiperplano separador óptimo.

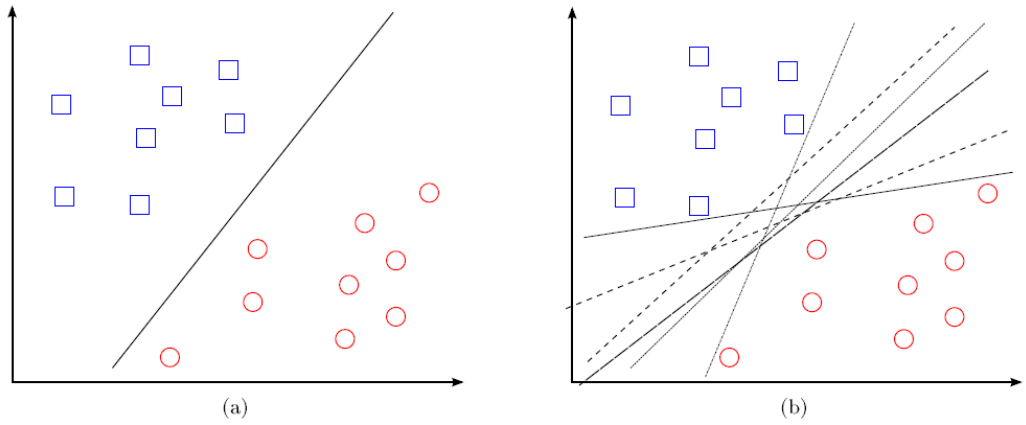

Figura 1.1: Hiperplanos de separación en un espacio bidimensional de un conjunto de ejemplos separables en dos clases: (a) un hiperplano de separación (b) algunos hiperplanos de separación, de entre los infinitos posibles. [Fuente: Figura 1 [C2014]]

Se define el **margen** τ, como la mínima distancia entre el hiperplano de separación y el ejemplo mas próximo a él de cualquiera de las dos clases. Vapnik y Lerner [VL1963] propusieron en 1963 tomar como hiperplano separador óptimo aquel que maximizara el margen. En tal caso, el hiperplano separador óptimo queda completamente caracterizado:

Proposición. *Un hiperplano separador se dirá óptimo si y solo si equidista del ejemplo más cercano de cada clase.*

Demostración. La demostración se puede hacer por reducción al absurdo. Supongamos que la distancia del hiperplano óptimo al ejemplo más cercano de la clase +1 fuese menor que la correspondiente al ejemplo más cercano de la clase -1. Esto significa que se puede alejar el hiperplano del ejemplo de la clase +1 una distancia tal que la distancia del hiperplano a dicho ejemplo sea mayor que antes y, a su vez, siga siendo menor que la distancia al ejemplo más cercano de la clase -1. Llegamos a contradicción, pues podemos aumentar el tamaño del margen cuando inicialmente suponíamos que éste era máximo. □

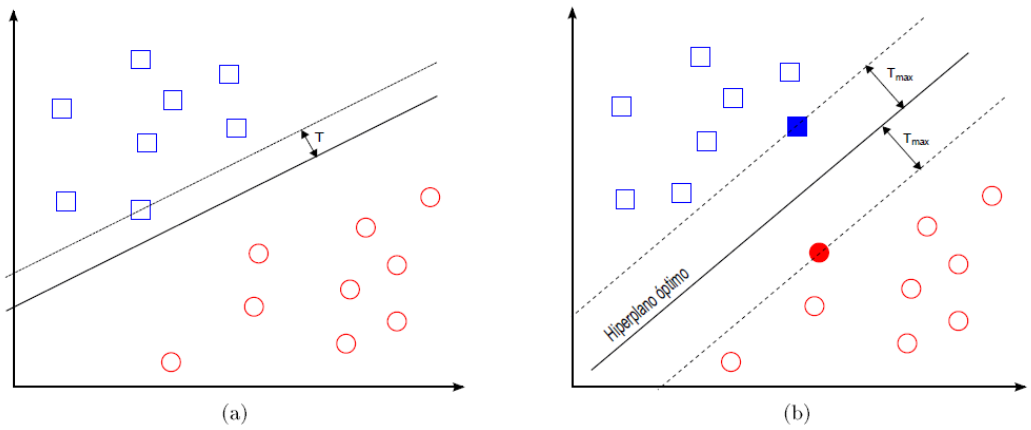

Figura 1.2: Margen de un hiperplano de separación: (a) hiperplano de separación no-óptimo y su margen asociado (no máximo) (b) hiperplano de separación óptimo y su margen asociado (máximo). [Fuente: Figura 2 [C2014]]

Veamos la formulación matemática de esta nueva restricción sobre el hiperplano separador: La distancia entre un hiperplano de separación $D(\mathbf{x})$ y un ejemplo $\mathbf{x}'$ viene dada por

$$\frac{|D(\mathbf{x}')|}{\|\mathbf{w}\|_2},$$

que junto con (1.2) obtenemos que todos los ejemplos de entrenamiento deben cumplir:

$$\frac{y_i D(\mathbf{x_i})}{\|\mathbf{w}\|_2} \geqslant \tau \qquad i = 1,\ldots,n. \tag{1.3}$$

De la expresión anterior se deduce que encontrar el hiperplano separador óptimo equivale a hallar el $\mathbf{w}$ que maximiza el margen. Puesto que existen infinitas soluciones que difieren solo en la escala de $\mathbf{w}$ se fija por convenio que

$$\tau\|\mathbf{w}\|_2 = 1.$$

Así llegamos a la conclusión de que maximizar el margen equivale a disminuir la norma de $\mathbf{w}$ y por tanto la condición (1.3) queda

$$y_i D(\mathbf{x_i}) \geqslant 1, \qquad i = 1,\ldots,n. \tag{1.4}$$

La búsqueda del hiperplano de separación óptimo se formaliza como un problema de optimización de tipo cuadrático:

$$\begin{array}{ll} \text{mín} & f(\mathbf{w}) = 1/2(\|\mathbf{w}\|_2)^2 = 1/2(\langle \mathbf{w},\mathbf{w}\rangle) \\ s.a & y_i(\langle \mathbf{w},\mathbf{x_i}\rangle + b) - 1 \geqslant 0, \qquad i = 1,\ldots,n. \end{array} \tag{1.5}$$

Observar que las restricciones de (1.2) quedan incluidas en (1.4). El hiperplano definido por $\mathbf{w}$ y b que minimizan el problema (1.5) recibe el nombre de **hiperplano de separación de margen duro** (hard margen). Resolver este problema de optimización cuadrática es difícil debido a la complejidad de las restricciones, por ello haremos uso de la teoría de optimización.

Un problema de optimización, primal, tiene una forma dual si la función a optimizar y las restricciones son estrictamente convexas. En ese caso resolver el problema dual es equivalente a obtener la solución del primal (véase Apéndice A). En el caso que nos ocupa se puede demostrar que tanto la función objetivo como las restricciones son funciones estrictamente convexas, por tanto admite un dual. Para hallarlo haremos uso de la función de Lagrange:

$$L(\mathbf{w},b,\alpha) = 1/2(\langle \mathbf{w},\mathbf{w}\rangle) - \sum_{i=1}^{n} \alpha_i(y_i(\langle \mathbf{w},\mathbf{x_i}\rangle + b) - 1), \qquad i = 1,\ldots,n, \tag{1.6}$$

donde $\alpha_i \geq 0$ son los multiplicadores de Lagrange. Aplicamos las condiciones de Karush-Kuhn-Tucker (KKT), derivando (1.6) respecto a las variables sobre las que optimizamos en el primal (primera condición de KKT) e igualando a cero los productos por los multiplicadores de Lagrange (condición complementaria):

$$\begin{array}{l} \dfrac{\partial L(\mathbf{w}^*,b^*,\alpha)}{\partial \mathbf{w}} \equiv \mathbf{w}^* - \displaystyle\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} = 0, \qquad i = 1,\ldots,n, \\ \dfrac{\partial L(\mathbf{w}^*,b^*,\alpha)}{\partial b} \equiv \displaystyle\sum_{i=1}^{n} \alpha_i y_i = 0, \qquad i = 1,\ldots,n, \end{array} \tag{1.7}$$

$$\alpha_i[(1 - y_i(\langle \mathbf{w}^*,\mathbf{x_i}\rangle + b^*)] = 0, \qquad i = 1,\ldots,n. \tag{1.8}$$

De (1.7) obtenemos la expresión de $\mathbf{w}^*$ en términos de los multiplicadores de Lagrange y sus restricciones,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x_i}, \qquad i = 1, \ldots, n, \tag{1.9}$$

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0, \qquad i = 1, \ldots, n. \tag{1.10}$$

Haciendo uso de (1.9) y (1.10) aplicados a (1.6) obtenemos la función a maximizar en el problema dual. Añadiendo las restricciones de no negatividad asociadas a los multiplicadores de Lagrange junto con (1.11) el problema dual es el siguiente:

$$\begin{aligned} \text{máx} \qquad & L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\langle \mathbf{x_i}, \mathbf{x_j} \rangle) \\ s.a \qquad & \sum_{i=1}^{n} \alpha_i y_i \geq 0, \qquad \alpha_i \geq 0, \; i = 1, \ldots, n. \end{aligned} \tag{1.11}$$

En general, la principal ventaja de resolver el problema dual es que el coste computacional es mucho menor, esto se debe a que en el problema dual el número de variables es directamente proporcional al tamaño de la muestra n, en cambio el problema primal lo es con la dimensionalidad de los ejemplos.

El vector direccional $\mathbf{w}^*$ del hiperplano óptimo lo obtenemos sustituyendo la solución del dual en (1.9)

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i (\langle \mathbf{x}, \mathbf{x_i} \rangle) + b^*, \qquad i = 1, \ldots, n. \tag{1.12}$$

Un ejemplo se dirá **separable** si satisface la restricción (1.4). Llamaremos **vectores soporte** (ver figura 1.3) a aquellos ejemplos que satisfacen (1.4) con igualdad. Caracterizamos estos ejemplos, para ello nos fijamos de nuevo en la condición complementaria de KKT (1.8) y se deduce que si en un ejemplo $\alpha_i > 0$ entonces

$$y_i(\langle \mathbf{w}^*, \mathbf{x_i} \rangle + b^*) = 1,$$

por consiguiente, se puede afirmar que sólo los ejemplos que tengan asociado un $\alpha_i > 0$ serán vectores soporte y por tanto son los únicos ejemplos que intervienen en la construcción del hiperplano. Además al ser los más cercanos al hiperplano de separación serán los mas difíciles de clasificar.

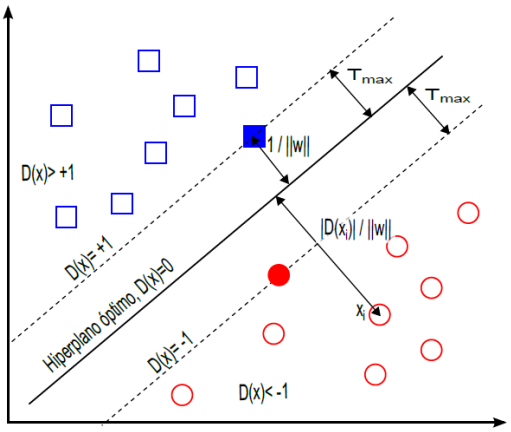

Figura 1.3: Distancia de cualquier ejemplo, $\mathbf{x_i}$, al hiperplano de separación óptimo, viene dada por $\frac{|D(\mathbf{x_i})|}{\|\mathbf{w}\|}$. En particular, si dicho ejemplo pertenece al conjunto de vectores soporte (identifcados por siluetas sólidas), la distancia a dicho hiperplano será siempre $\frac{1}{\|\mathbf{w}\|}$. Además, los vectores soporte aplicados a la función de decisión siempre cumplen que $|D(\mathbf{x_i})| = 1$. [Fuente: Figura 3 [C2014]]

La determinación de b^* la realizamos a partir de (1.8), de tal forma que si $\alpha_i > 0$, es decir, estamos con un vector soporte del problema, en ese caso

$$y_i(\langle \mathbf{w}^*, \mathbf{x_i} \rangle + b^*) = 1,$$

y si despejamos b^*, obtenemos

$$b^* = y_{vs} - \langle \mathbf{w}^*, \mathbf{x_{vs}} \rangle,$$

donde $(\mathbf{x_{vs}}, y_{vs})$ representa la tupla de cualquier ejemplo que satisfaga la igualdad anterior, es decir que sea vector soporte. En la práctica es más robusto obtener b^* promediando a partir de todos los vectores soporte, sea vs dicho conjunto de índices y N_{vs} su cardinal, en tal caso

$$b^* = \frac{1}{N_{vs}} \sum_{i \in vs} (y_i - \langle \mathbf{w}^*, \mathbf{x_i} \rangle).$$

1.2. Ejemplos cuasi-separables linealmente

En la realidad es muy frecuente disponer de ejemplos que no son perfecta y linealmente separables. Cortes y Vapnik [CV1995] demuestran en 1995 que este tipo de problemas son mejores de tratar si permitimos que algunos ejemplos no verifiquen las restricciones sobre el margen formuladas en el problema primal (1.5). Dicho de otra forma, podemos relajar el grado de separabilidad entre los conjuntos de ejemplos permitiendo que haya errores de clasificación en algunos ejemplos del conjunto de entrenamiento. En este contexto se pueden dar dos situaciones para un ejemplo: cae dentro del margen asociado a la clase correcta, de acuerdo a la frontera de decisión que define el hiperplano de separación y en el otro caso, el ejemplo cae al otro lado de dicho hiperplano. En ambos casos se dice que el **ejemplo** es **no-separable**, pero en el primer caso es clasificado de forma correcta y en el segundo, no lo es.

Para abordar este problema introducimos un conjunto de variables reales positivas $\xi_i, i = 1, \ldots n$ llamadas **variables de holgura**, cada una asociada a un ejemplo, de forma que podemos controlar el numero de ejemplos no separables. Las restricciones relajadas que deberá verificar cada ejemplo del conjunto de entrenamientos serán:

$$y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b^*) \geqslant 1 - \xi_i, \qquad i = 1, \ldots, n. \tag{1.13}$$

Podemos entender la **variable de holgura** asociada a cada ejemplo $(\mathbf{x_i}, y_i)$ como la desviación de la situación separable, medida desde el margen de la clase correspondiente a dicho ejemplo. Estas variables representan el potencial incumplimiento de las restricciones sobre el margen para cada ejemplo. Los ejemplos se pueden clasificar (ver figura 1.4) en separables si su variable de holgura correspondiente toma valor cero, si toma valor entre cero y uno, no separables pero correctamente clasificados y si es mayor que uno son ejemplos no separables y mal clasificados. Por tanto es fácil deducir que $\sum_{i=1}^{n} \xi_i$ permite medir el coste asociado al numero de ejemplos no-separables, cuanto mayor sea el valor de esta suma, mayor sera el número de ejemplos no separables.

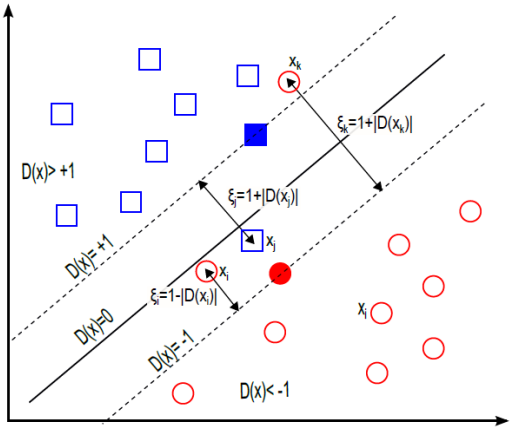

Figura 1.4: Los ejemplos $\mathbf{x_i}$, $\mathbf{x_j}$ y $\mathbf{x_k}$ son, cada uno de ellos, no separables ($\xi_i, \xi_j, \xi_k > 0$). Sin embargo, $\mathbf{x_i}$ está correctamente clasificado, mientras que $\mathbf{x_j}$ y $\mathbf{x_k}$ están en el lado incorrecto de la frontera de decisión y, por tanto, mal clasificados. [Fuente: Figura 4 [C2014]]

En el caso que nos ocupa la función a optimizar debe incluir los errores de clasificación que comete el hiperplano de separación, es decir:

$$f(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \tag{1.14}$$

donde C es una constante suficientemente grande, cuyo valor se determina manualmente y que permite controlar en qué grado influye el término del coste de ejemplos no-separables en la minimización de la norma, es decir, permitirá regular el compromiso entre el grado de sobreajuste del clasificador final y la proporción del número de ejemplos no separables (complejidad y error empírico o de entrenamiento). La elección de una C u otra determinará en cierto modo la calidad del clasificador como veremos en el último capítulo. En consecuencia el nuevo problema de optimización es:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.a \quad & y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b^*) + \xi_i - 1 \geq 0, \\ & \xi_i \geq 0, \qquad i = 1,\ldots,n. \end{aligned} \tag{1.15}$$

El hiperplano de separación, definido por $\mathbf{w}$ y b, resultante de resolver el problema (1.15) se denomina **hiperplano de separación de margen blando** (soft margin).

Vapnik y Chervonenkis [VCh1989] demostraron en 1989 que el aumento de la complejidad de la función separadora provoca una disminución del error de clasificación del conjunto de entrenamientos sin embargo esto nos lleva a un sobreentrenamiento de la SVM de forma que aumenta el riesgo de error cuando se aplica sobre ejemplos de prueba, es decir se pierde generalidad. Podemos ver la expresión a minimizar en 1.15 como un compromiso entre la complejidad de la SVM inversamente proporcional a $\|\mathbf{w}\|$ y el sobreentrenamiento controlado por la suma de las variables de holgura.

La resolución del problema primal (1.15) es similar al caso separable de la sección anterior con la principal diferencia de que al tener dos familias de restricciones en (1.15) aparecerán dos familias de multiplicadores de Lagrange. La resolución detallada se puede consultar en [C2014]. El problema dual

que se obtiene es:

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \\ s.a \quad & \sum_{i,=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, n. \end{aligned} \tag{1.16}$$

Observar que ahora tenemos una clasificación aún más completa, que en el caso separable, de los vectores soportes dado que los multiplicadores de Lagrange α_i tienen cota superior. El razonamiento (ver [C2014]) discurre paralelamente al de la obtención de una expresión para el calculo de b^* y se deduce principalmente del criterio de optimalidad (1.13) y de aplicar las KKT al problema primal.

Tenemos dos tipos de ejemplos de entrenamiento para los que $\alpha_i > 0$, luego satisfacen la restricción (1.13) con igualdad, es decir, son vectores soporte y por tanto intervienen en la construcción del hiperplano separador óptimo:

- si $0 < \alpha_i^* < C$ se tiene que $\xi_i = 0$ por tanto son ejemplos separables. Se denotan **vectores soporte "normales"** (free support vectors).
- si $\alpha_i^* = C$, entonces $0 < \xi_i$ luego son ejemplos no separables y clasificados correctamente si $0 < \xi_i < 1$ o mal clasificados si $\xi_i > 1$. Reciben el nombre de **vectores soporte acotados** (bounded support vectors).

Los ejemplos de entrenamiento tales que $\alpha_i = 0$, no son vectores soporte. Además, verifican $\xi_i = 0$ y por tanto corresponden a los ejemplos separables.

1.3. Ejemplos no separables linealmente

Cuando el conjunto de ejemplos no se puede separar por medio de una función lineal, un hiperplano separador, recurrimos a una nueva técnica consistente en la transformación del espacio original mediante una función no lineal hacia un espacio Hilbert dotado de un producto escalar denominado función kernel (ver figura 1.5). Llamaremos **espacio de entradas** al espacio original de los ejemplos **x**. El espacio transformado (de alta dimensionalidad) se llama **espacio de características** y se definirá a partir de un conjunto de funciones base no lineales.

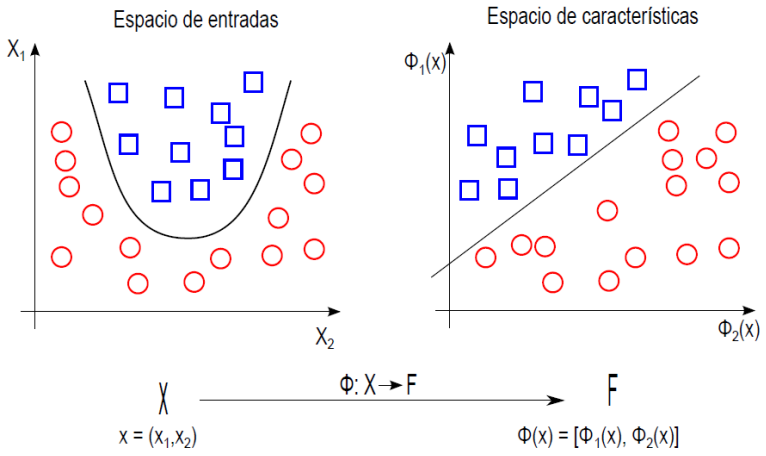

Figura 1.5: El problema de la búsqueda de una función de decisión no lineal en el espacio de entradas, se puede transformar en un nuevo problema consistente en la búsqueda de una función de decisión lineal (hiperplano) en un nuevo espacio transformado, el espacio de características. [Fuente: Figura 5 [C2014]]

Sea $\Phi : \mathbb{X} \longrightarrow \mathscr{F}$ la función de transformación que hace corresponder a cada vector de entrada $\mathbf{x}$ con un punto en el espacio de características $\mathscr{F}$, donde $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]$ y $\exists\ \phi_i(\mathbf{x})$, $i = 1, \ldots, m$ tal que $\phi_i(\mathbf{x})$ es una función no lineal. Por definición una función **kernel** es una función $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ que a cada par de elementos del espacio de entrada $\mathbb{X}$, le asigna, un valor real correspondiente al producto escalar de las imágenes de dichos elementos en el espacio de las características $\mathscr{F}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = \phi_1(\mathbf{x})\phi_1(\mathbf{x}') + \cdots + \phi_m(\mathbf{x})\phi_m(\mathbf{x}'). \tag{1.17}$$

La teoría de Espacios de Hilbert con Núcleo Reproductor [A1944], muestra que las funciones Kernel se corresponden con un producto escalar y que este induce un espacio lineal con mayor dimensión que el espacio original, posiblemente infinita. La respuesta de como transformar el espacio de entradas, de dimensión finita, en otro espacio de dimensión infinita nos la da el siguiente teorema.

Teorema (de Aronszajn). *Para cualquier función $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ que sea simetrica*[1] *y semidefinida positiva*[2]*, existe un espacio Hilbert y una función $\Phi : \mathbb{X} \longrightarrow \mathscr{F}$ tal que*

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \qquad \forall\ \mathbf{x}, \mathbf{x}' \in \mathbb{X}. \tag{1.18}$$

Boser, Guyon y Vapnik [BGV1992] demostraron en 1992 que como consecuencia de este teorema para construir una función kernel no es necesario hacerlo a partir de un conjunto de funciones base $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]$, simplemente basta definir una función que cumpla las dos condiciones del teorema. De esta forma el kernel representa el producto escalar $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ que induce un espacio de alta dimensionalidad. Por tanto, para evaluar una función kernel no se necesitara conocer dicho conjunto de funciones base.

Este hecho nos permite reproducir cualquier algoritmo lineal en un espacio de Hilbert o equivalentemente para cualquier algoritmo existe una versión no lineal, donde la transformación (no lineal) es Φ. Este hecho se conoce como el truco de los kernels.

Algunos ejemplos de funciones kernel son:

- Kernel lineal: $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$.
- Kernel polinómico de grado-p: $K_p(\mathbf{x}, \mathbf{x}') = [\gamma \langle \mathbf{x}, \mathbf{x}' \rangle + \tau]^p$.
- Kernel gaussiano: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2),\ \gamma > 0$. En la práctica, cualquier separador puede construirse usando este kernel.
- Kernel sigmoidal: $K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \langle \mathbf{x}, \mathbf{x}' \rangle + \tau)$.

A los parámetros γ, τ y p se les denomina parámetros del kernel.

Una caracterización de las funciones kernel la proporciona el **teorema de Mercer** [J1982], que nos asegura que toda función $K(\mathbf{u}, \mathbf{v})$ que verifica

$$\int_{uv} K(\mathbf{u}, \mathbf{v}) g(u) g(v)\, du\, dv > 0,$$

para toda función $g(.)$ de cuadrado integrable, es una función kernel.

[1] Una función $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ es simetrica si $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}) \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{X}$

[2] Una función $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ es semidefinida positiva si $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x_i}, \mathbf{x_j}) \geq 0$ para cualesquiera conjuntos $\mathbf{x_1}, \ldots, \mathbf{x_n} \in \mathbb{X}$ y $c_1, \ldots, c_n \in \mathbb{R}$, siendo $n > 0$

Volvemos a la formulación del problema en el caso no separable linealmente. La idea es construir un hiperplano de separación lineal en el espacio de las características. La frontera de decisión lineal obtenida en el nuevo espacio se transformará en una frontera de decisión no lineal en el espacio original de entradas. En este contexto el hiperplano separador en el espacio de características viene dada por

$$D(\mathbf{x}) = w_1\phi_1(\mathbf{x}) + \cdots + w_m\phi_m(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle. \tag{1.19}$$

Observar que omitimos el parámetro b dado que se puede incluir en la base de funciones de transformación con la función constante $\phi_1(\mathbf{x}) = 1$.

El planteamiento del problema es igual que en la sección anterior, basta sustituir en el dual (1.16) el producto escalar por la función Kernel:

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \\ s.a \quad & \sum_{i,=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, n. \end{aligned} \tag{1.20}$$

La razón por la que, ahora, el problema de optimización se expresa sólo en su forma dual, es la posible dimensionalidad infinita del espacio de características ya que la solución del dual no depende de la dimensionalidad del espacio sino de la cardinalidad del conjunto de ejemplos.
La función de decisión vendrá dada por:

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x_i}), \qquad i = 1, \ldots, n, \tag{1.21}$$

donde el valor de los parámetros $\alpha_i, i = 1, \ldots n$ se obtendrán como solución al problema de optimización cuadrática dado por (1.20). Como ya mencionamos, no existe una forma teórica de encontrar el valor del parámetro de regularización C. Sólo existe la heurística de usar un valor grande. La forma de determinar C y el resto de parámetros del kernel se basa en técnicas de **validación cruzada** como veremos en el último capítulo.

1.3.1. Solución del problema OR-exclusivo mediante SVMs

El **problema or-exclusivo** pertenece al caso de problemas separables no-linealmente. Se define como el problema de encontrar un hiperplano de separación que clasifique sin error los siguientes 4 ejemplos:

Ejemplo	$\mathbf{x} = (x_1, x_2)$	y
1	$(+1, +1)$	$+1$
2	$(-1, +1)$	-1
3	$(-1, -1)$	$+1$
4	$(+1, -1)$	-1

Cuadro 1.1: ejemplos del problema OR-exclusivo

La resolución que se propone para el problema es crear un clasificador SVM usando un Kernel polinómico con $p = 2, \gamma = 1$ y $\tau = 1$, $K_2(\mathbf{x}, \mathbf{x}') = [\langle \mathbf{x}, \mathbf{x}' \rangle + 1]^2$

Una vez resuelto el problema dual

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{4} \alpha_i \alpha_j y_i y_j K_2(\mathbf{x}, \mathbf{x_i}) \\ s.a \quad & \sum_{i,=1}^{4} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, 4, \end{aligned} \tag{1.22}$$

la solución queda $\alpha_i = 0{,}125$, $i = 1, \ldots 4$ y la funcion de decisión lineal $D(\mathbf{x}) = 0{,}125 \sum_{i=1}^{4} y_i K_2(\mathbf{x}, \mathbf{x_i})$. Dado que ningún i cumple $\alpha_i^* = 0$, podemos afirmar que todos los ejemplos son vectores soporte. Hagamos un estudio de la función kernel y el conjunto de funciones base (las operaciones se pueden consultar en [C2014]).

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = K_2(\mathbf{x}, \mathbf{x}') =$$
$$= [\langle \mathbf{x}, \mathbf{x}' \rangle + 1]^2 = \langle (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, {x_1}^2, {x_2}^2), (1, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2', {x_1'}^2, {x_2'}^2) \rangle.$$

La base de funciones de transformación al espacio de características es:

$$\phi_1(x_1, x_2) = 1, \phi_2(x_1, x_2) = \sqrt{2}x_1, \phi_3(x_1, x_2) = \sqrt{2}x_2, \phi_4(x_1, x_2) = \sqrt{2}x_1x_2, \phi_5(x_1, x_2) = {x_1}^2 \phi_6(x_1, x_2) = {x_2}^2.$$

Finalmente, la función de decisión puede expresarse como

$$D(\mathbf{x}) = 0{,}125 \sum_{i=1}^{4} y_i K_2(\mathbf{x}, \mathbf{x_i}) = \frac{1}{\sqrt{2}} \phi_4(\mathbf{x}).$$

A partir de esto podemos afirmar que de las seis dimensiones del espacio de características en el que se ha transformado el espacio bidimensional original, la función de decisión lineal se expresa en términos de sólo una ϕ_4. Basta una dimensión del espacio transformado para separar los 4 ejemplos del conjunto de entrenamiento original (ver figura 1.6). Es fácil comprobar (ver [C2014]) que $\phi_4(\mathbf{x}) = 0$ es el hiperplano de separación en el espacio de características y $\phi_4(\mathbf{x}) = \sqrt{2}, \phi_4(\mathbf{x}) = -\sqrt{2}$ delimitan los margenes. En el espacio original la función de decisión no lineal es $x_1x_2 = 0$ y los márgenes son las hipérbolas $x_1x_2 = 1$, $x_1x_2 = -1$.

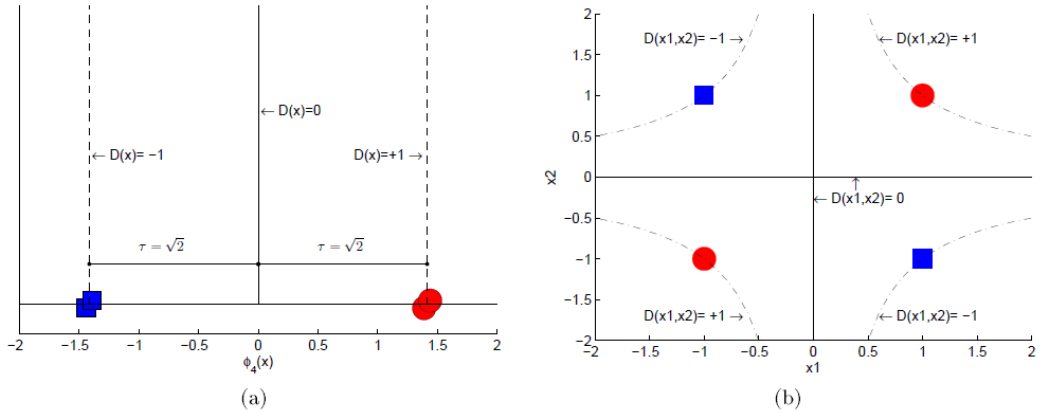

Figura 1.6: Solución al problema XOR: (a) hiperplano de separación en el espacio de características, junto con su margen asociado (los cuatro ejemplos son vectores soporte) (b) función de decisión no lineal en el espacio de ejemplos original resultante de transformar el hiperplano obtenido en (a) en coordenadas del espacio original.[Fuente: Figura 7 [C2014]]

Capítulo 2

Clasificación multinomial y regresión

En este capítulo vamos a introducir dos extensiones de las SVM especialmente útiles en la práctica: la clasificación en más de dos categorías y la predicción de una respuesta continua, es decir, un modelo de regresión.

2.1. Caso multinomial

Las SVM fueron inicialmente diseñadas para clasificación binaria. Sin embargo los problemas reales a menudo requieren discernir en mas de dos categorías.

Para extender SVM al caso de clasificación múltiple han sido propuestos muchos modelos. A grandes rasgos podemos diferenciar las técnicas existentes en métodos directos e indirectos.

2.1.1. Métodos indirectos

Los problemas de clasificación múltiple se suelen descomponer en una sucesión de problemas binarios de forma que podemos aplicar el método estándar de resolución de SVM a cada uno por separado. Los dos métodos más representativos de esta técnica son uno frente al resto (1VR) y uno frente a uno (1V1). Ambos métodos son casos particulares de los Códigos Correctores de Errores de Salida (ECOC) [BD1995], los cuales, descomponen el problema multiclase en un conjunto de problemas binarios.

Uno frente al resto (one-versus-rest)

Suponer que tenemos un problema de clasificación con k clases diferentes, el método 1VR construye (entrena) k hiperplanos separadores. Para construir el m-ésimo clasificador binario, dado por $f_m(\mathbf{x}) = \mathbf{w_m}^T \Phi(\mathbf{x}) + b_m$ se consideran los ejemplos de entrenamiento de la m-ésima clase como de tipo positivo y los restantes ejemplos de las $k-1$ clases como negativos. De entre los k clasificadores el que nos proporciona el máximo valor de $f_i(\mathbf{x_{new}})$, $i \in 1,\ldots,k$ es elegido para determinar la etiqueta (clase) del ejemplo de prueba $\mathbf{x_{new}}$. El principal problema de la técnica uno frente al resto es el desequilibrio del conjunto de entrenamientos: si suponemos que de cada clase hay el mismo numero de ejemplos de entrenamientos la ratio de ejemplos positivos frente a negativos sera de $\frac{1}{k-1}$, por tanto la simetría del problema original se pierde.

Uno frente a uno (one-versus-one)

Dada una clase el método uno frente a uno estudia todos los posibles pares de clases, entre la dada y cada una de las restantes induciendo un clasificador para cada uno de ellos. Se generan por tanto $\frac{k(k-1)}{2}$

clasificadores. Consideramos un ejemplo de prueba $\mathbf{x_{new}}$ y evaluamos cada clasificador, comparamos ambas clases y a la clase ganadora se le da un voto, el ejemplo de prueba se clasificará en la clase que más votos tenga. El número de clasificadores creados con este método es mucho mayor que con el anterior, sin embargo en el problema dual de optimización cuadrática el número de variables escala en cada caso con el número de ejemplos y consideramos dos clases, por tanto el tamaño del problema es mucho menor que en el caso 1VR. Esto hace posible un entrenamiento más rápido. Para decidir la categoría de cada ejemplo probamos $k-1$ hiperplanos.

2.1.2. Métodos directos

Los métodos directos de clasificación múltiple basados en SVM enfocan el problema a un único proceso de optimización, a través de combinar problemas de clasificación binaria en una única función objetivo, de forma que se logra simultáneamente una clasificación de todas las clases. Sin embargo esto conlleva una mayor complejidad a nivel computacional debido al tamaño del problema de optimización cuadrática resultante.

SVM multinomial de Weston y Watkin

Vapnik [V1998], Weston y Watkin [WW1999] proponen que para un problema de clasificación de k-clases se diseñe una única función objetivo que entrene las k SVMs binarias simultáneamente y maximice los márgenes de cada cada clase con el resto de ejemplos. A grandes rasgos, es la idea del método uno frente al resto unificado en una sola función objetivo. Dado un conjunto de l ejemplos $S = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_l}, y_l)\}$, $\mathbf{x_i} \in \mathbb{R}^d$ correspondientes a k clases distintas, $y_i \in \{1, \ldots, k\}$, la formulación general del problema primal en el espacio de características es:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\sum_{m=1}^{k} \|\mathbf{w_m}\|^2 + C\sum_{i=1}^{l}\sum_{t\neq y_i} \zeta_{i,t} \\ s.a \quad & \mathbf{w_{y_i}}^T \Phi(\mathbf{x_i}) + b_{y_i} \leq \mathbf{w_{y_i}}^T \Phi(\mathbf{x_i}) + b_t + 2 - \zeta_{i,t}, \\ & \zeta_{i,t} \geq 0, \qquad i = 1, \ldots, l \ , \ t \in 1, \ldots, k \setminus y_i, \end{aligned} \tag{2.1}$$

donde $\mathbf{w_m}$ son cada uno de los vectores direccionales del hiperplano separador en el espacio Hilbert correspondiente para cada problema de clasificación binario, $\mathbf{b} \in \mathbb{R}^k$ son los términos independientes para cada uno de los k clasificadores lineales, que se almacenan en un vector y la matriz $\zeta \in \mathbb{R}^{l \times k}$ las variables de holgura. Las variables de holgura $\zeta_{i,m} = 1 - f_{y_i}(\mathbf{x_i}) + f_m(\mathbf{x_i})$, $m \in 1, \ldots, k \setminus y_i$, si son positivas representan la desviación respecto al margen correcto entre la clase i y otra clase m (ver figura 2.1). Considerar la función $[.]_+ \equiv max(0, .)$, definimos la **función de pérdida** como $\xi_i^{(1)} = \sum_{m \neq y_i} [\zeta_{i,m}]_+$. La función de decisión resultante es:

$$\underset{m}{\text{argmax}}\, f_m(\mathbf{x}) = \underset{m}{\text{argmax}}(\mathbf{w_m}^T \Phi(\mathbf{x}) + b_m).$$

La principal desventaja de este método es que el tiempo computacional puede ser muy grande dado la enorme cantidad de variables en el problema de optimización cuadrática.

SVM multinomial de Crammer y Singer

Crammer y Singer [CS2001] presentaron en 2001 el siguiente plantamiento:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\sum_{m=1}^{k}\|\mathbf{w_m}\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.a} \quad & \mathbf{w_{y_i}}^T\Phi(\mathbf{x_i}) - \mathbf{w_{y_i}}^T\Phi(\mathbf{x_i}) \leq 1 - \delta_{y_i,t} - \xi_i, \\ & i = 1,\ldots,l,\ t \in 1,\ldots,k, \end{aligned} \tag{2.2}$$

donde $\delta_{i,j}$ es la delta de Kronecker, y $\xi \in \mathbb{R}^l$. Notar que las restricciones $\xi_i \geq 0$, $i = 1,\ldots,l$ están implícitamente en (2.2) cuando $t = y_i$. En este caso la función de pérdida se define como la mayor de las variables de holgura, es decir, ${\xi_i}^{(2)} = [\text{máx}_{m\neq y_i}\,\zeta_{i,m}]_+$. La función de decisión es:

$$\underset{m}{\text{argmax}}\, f_m(\mathbf{x}) = \underset{m}{\text{argmax}}(\mathbf{w_m}^T\Phi(\mathbf{x})).$$

Aunque este método nos de un conjunto compacto de restricciones, el número de variables en el problema dual es todavía elevado $l \times k$.

La siguiente técnica se basa en estos algoritmos [CS2001] para desarrollar un método mas simplificado llamado SimMSVM, mediante la relajación de las restricciones. Dicho de otro modo resolver un único problema de programación cuadrática de l variables es suficiente para resolver un problema de clasificación multiclase.

SimMSVM

Con el objetivo de reducir el tamaño del problema, el número de restricciones debe ser proporcional al número de ejemplos l en lugar de $l \times k$. Para construir el clasificador multiclase de SimMSVM introducimos las siguientes funciones de pérdida relajadas,

$${\xi_i}^{(3)} = [\frac{1}{k-1}\sum_{m=1,m\neq y_i}^{k}\zeta_{i,m}]_+.$$

La pérdida sufrida es linealmente proporcional a la media de las distancias entre el ejemplo y el margen correspondiente a su clase. La relación entre las funciones de pérdida ${\xi_i}^{(1)}, {\xi_i}^{(2)}, {\xi_i}^{(3)}$ es la que sigue:

$${\xi_i}^{(1)} \geq {\xi_i}^{(2)} \geq {\xi_i}^{(3)}.$$

Es de esperar que la función de pérdida relajada ${\xi_i}^{(3)}$ incurra en mas errores de clasificación (recordar capítulo 1) en el sentido de que la pérdida para cada ejemplo puede no estar entre 0 y 1, y por tanto el ejemplo estar mal clasificado. Sin embargo en la práctica estamos dispuestos a admitir cierto numero tolerable de errores de clasificación pues de esta forma obtenemos una velocidad significativa durante el proceso de entrenamiento.

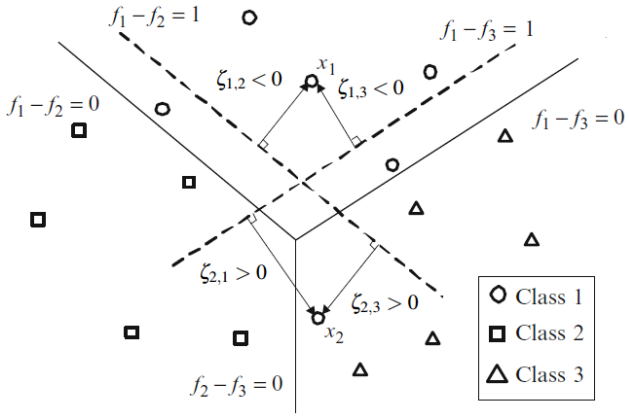

Figura 2.1: Clasificación multiclase. Representamos tres clases con círculos, rectángulos y triángulos. Las lineas de trazo grueso representan posibles frontera de decisión. Las lineas punteadas son margenes positivos para cada par de clases delimitadas por la frontera de decisión. Observar que para un ejemplo correctamente clasificado como x_1 tenemos que $\xi_i^{(1)} = \xi_i^{(2)} = \xi_i^{(3)} = 0$ puesto que las variables de holgura son negativas $\zeta_{1,2} < 0$, $\zeta_{1,3} < 0$. En el caso opuesto el ejemplo x_2 esta mal clasificado y viola las dos fronteras de los margenes $\zeta_{2,1} > 0$, $\zeta_{2,3} > 0$. Los tres métodos generan una pérdida [Modificado de Fuente: Figura 2.1 [WX2014]]

Tras la aplicación de las nuevas funciones de pérdida, el problema (2.2) de Crammer y Singer queda:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\sum_{m=1}^{k} \|\mathbf{w_m}\|^2 + C\sum_{i=1}^{l} \xi_i \\ s.a \quad & \mathbf{w_{y_i}}^T \Phi(\mathbf{x_i}) - \frac{1}{k-1}\sum_{m \neq y_i} \mathbf{w_m}^T \Phi(\mathbf{x_i}) \leq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{2.3}$$

Para transformar el problema a su dual usamos los argumentos habituales de KKT (ver capítulo 2 de [WX2014]). La función de decisión resultante es

$$\underset{m}{\text{argmax}}\, f_m{}^*(\mathbf{x}) = \underset{m}{\text{argmax}} \sum_{i:y_i=m} \alpha_i{}^* K(\mathbf{x_i}, \mathbf{x}),$$

donde $K(\mathbf{x_i}, \mathbf{x_j})$ es el elemento i,j de la matriz kernel.

Los métodos propuestos por Weston y Watkins, Crammer y Singer satisfacen el rango [0,1] de la función de pérdida. Sin embargo en la práctica son de una gran complejidad a nivel computacional debido al gran tamaño del problema de optimización cuadrática cuyo número de variables es el producto del numero de ejemplos l por el numero de clases k. SimMSVM reduce el tamaño del problema dual de $l \times k$ a l variables introduciendo una cota del error de clasificación relajada. En conjuntos de datos reales SimMSVM alcanza una velocidad satisfactoria durante el proceso de entrenamiento, al mismo tiempo que mantiene a la hora de clasificar una exactitud muy competitiva frente a los otros métodos de resolución de problemas multiclase.

2.2. SVM para regresión

En esta sección estudiaremos la aplicaciónde las SVM en la resolución de problemas de regresión. En estos casos, suelen llamarse SVR (Support Vector Regression).

Dado un conjunto de ejemplos de entrenamiento $S = \{(\mathbf{x_1}, y_1) \dots (\mathbf{x_n}, y_n)\}$, donde $\mathbf{x_i} \in \mathbb{R}^d$ e $y_i \in \mathbb{R}$, en el que se asume que los valores y_i de todos los ejemplos de S se pueden ajustar (o cuasi-ajustar) mediante un hiperplano, nuestro objetivo es encontrar los parámetros $\mathbf{w} = (w_1, \dots, w_d)$ que permitan definir el hiperplano de regresión $f(\mathbf{x}) = (w_1 x_1 + \cdots + w_d x_d) + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$.

Definimos el **ruido o perturbación aleatoria** $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma)$ como el error en la medición del valor y, es decir $y = f(\mathbf{x}) + \epsilon$. Para permitir cierto ruido en los ejemplos de entrenamiento se puede relajar la condición de error entre el valor predicho por la función y el valor real. Para esto se utiliza la función de perdida ϵ-insensible, L_ϵ, definida por:

$$L_\epsilon(\mathbf{x}) = \begin{cases} 0 & \text{si } |y - f(\mathbf{x})| \leq \epsilon, \\ |y - f(\mathbf{x})| - \epsilon & \text{en otro caso.} \end{cases} \tag{2.4}$$

Es una función lineal con una zona insensible de anchura 2ϵ, en la que la función de pérdida toma valor nulo. Eligiendo esta función permitimos cierta flexibilidad en la función solución, de forma que todos los ejemplos que quedan confinados en la región tubular no serán considerados vectores soporte, pues como hemos visto el coste asociado a la función de pérdida es 0. En la práctica es muy difícil lograr un modelo de regresión lineal con error de predicción cero por ello recurriremos al concepto de margen blando introducido en el capitulo 1.

Definimos las variables de holgura como la distancia al ejemplo medida desde la zona tubular del hiperplano de regresión. Las variables de holgura ${\xi_i}^+$ y ${\xi_i}^-$ permitirán cuantificar el error de predicción que se esta dispuesto a admitir para cada ejemplo de entrenamiento y con la suma de todas ellas el coste asociado a los ejemplos con un error de predicción no nulo. Tomaremos ${\xi_i}^+ > 0$ cuando la predicción del ejemplo $f(\mathbf{x_i})$ es mayor que su valor real, y_i, en una cantidad superior a ϵ equivalentemente $f(\mathbf{x_i}) - y_i > \epsilon$. Análogamente ${\xi_i}^- > 0$ cuando el valor real del ejemplo es mayor que su predicción en una cantidad superior a ϵ, es decir, $y_i - f(\mathbf{x_i}) > \epsilon$. En cualquier otro caso las variables de holgura toman valor cero. Notar que ambas variables no pueden tomar simultáneamente valor distinto de cero, ocurre siempre que ${\xi_i}^+ {\xi_i}^- = 0$ (ver figura 2.2).

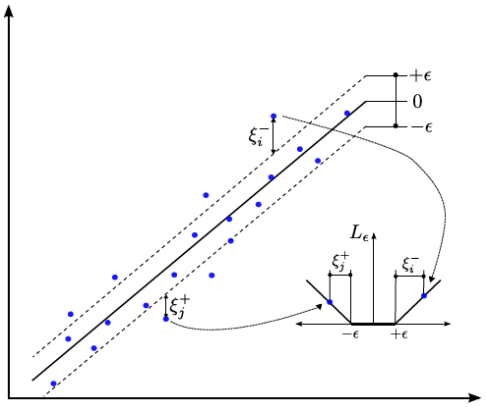

Figura 2.2: SVR con margen blando: se muestra la relación entre las variables de holgura, ${\xi_i}^-$,${\xi_j}^+$, asociadas a ejemplos que quedan fuera de la zona tubular ϵ-insensible y la función de pérdida, L_ϵ. [Fuente: Figura 8 [C2014]]

Con todo esto ya podemos plantear el problema a optimizar. Nuestro objetivo es minimizar la suma de las funciones de pérdida asociadas, cada una a un ejemplo del conjunto de entrenamientos [1] $\sum_{i=1}^{n} L_\epsilon(y_i, f(\mathbf{x_i})) = \sum_{i\in\text{zona no tubular}} |y_i - f(\mathbf{x_i})| - \epsilon$. Esto es equivalente a maximizar la zona tubular definida por la función de pérdida, en la cual esta toma valor nulo[2] por tanto maximizar ϵ equivale a minimizar $\|\mathbf{w}\|$. Todo ello unido a la penalización impuesta por las variables de holgura definen un problema de optimización análogo al problema de clasificación con margen blando (1.15) con la salvedad de que ahora tenemos dos tipos de variables de holgura:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i{}^+ + \xi_i{}^-) \\ s.a \quad & (\langle \mathbf{w}, \mathbf{x_i}\rangle + b) - y_i - \epsilon - \xi_i{}^+ \leq 0, \\ & y_i - (\langle \mathbf{w}, \mathbf{x_i}\rangle + b) - \epsilon - \xi_i{}^- \leq 0, \\ & \xi_i{}^+, \xi_i{}^- \geq 0, \qquad i = 1, \ldots, n. \end{aligned} \tag{2.5}$$

La transformación al problema dual (ver [C2014]) es análoga a las vistas hasta ahora con la diferencia de que consideraremos cuatro familias de multiplicadores de Lagrange: $\alpha_i{}^+, \alpha_i{}^-, \beta_i{}^+, \beta_i{}^-$.

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{n}(\alpha_i{}^- - \alpha_i{}^+)y_i - \epsilon\sum_{i=1}^{n}(\alpha_i{}^- + \alpha_i{}^+) - \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i{}^- - \alpha_i{}^+)(\alpha_j{}^- - \alpha_j{}^+)\langle \mathbf{x_i}, \mathbf{x_j}\rangle \\ s.a \quad & \sum_{i=1}^{n}(\alpha_i{}^+ - \alpha_i{}^-) = 0, \\ & 0 \leq \alpha_i{}^+,\ \alpha_i{}^- \leq C,\ i = 1, \ldots, n. \end{aligned} \tag{2.6}$$

El regresor obtenido es:

$$\sum_{i=1}^{n}(\alpha_i{}^- - \alpha_i{}^+)\langle \mathbf{x}, \mathbf{x_i}\rangle + b^*. \tag{2.7}$$

El valor óptimo de b^* se obtiene (ver [C2014]) a partir de las restricciones resultantes de la aplicación de la segunda condición KKT y las restricciones sobre el dual, de forma que:

$$\begin{aligned} b^* &= y_i - \langle \mathbf{w}^*, \mathbf{x_i}\rangle + \epsilon, \qquad \text{si } 0 < \alpha_i{}^+ < C, \\ b^* &= y_i - \langle \mathbf{w}^*, \mathbf{x_i}\rangle - \epsilon, \qquad \text{si } 0 < \alpha_i{}^- < C. \end{aligned}$$

Observar que para definir el hiperplano de regresión tenemos en cuenta los ejemplos con función de pérdida no nula, es decir, los que se encuentran en el exterior de la región tubular. Visto en términos de los parámetros introducidos anteriormente, para los vectores soporte se deduce de las condiciones KKT que $\alpha_i{}^+\alpha_i{}^- = 0$, entonces

- para los ejemplos que están fuera de la zona tubular se cumplirá $\xi_i{}^+\xi_i{}^- = 0$, si $\xi_i{}^- = 0$ y $\xi_i{}^+ > 0$ entonces $\alpha_i{}^+ = C$ y $\alpha_i{}^- = 0$ y si $\xi_i{}^- > 0$ y $\xi_i{}^+ = 0$ entonces $\alpha_i{}^- = C$ y $\alpha_i{}^+ = 0$.
- los vectores soporte que caen justo en la frontera de la zona de sensibilidad verifican que si $0 < \alpha_i{}^+ < C$ entonces $\alpha_i{}^- = 0$, en ese caso debe ser $\xi_i{}^+ = 0$ y $\xi_i{}^- = 0$. Análogamente para el otro caso.

[1] En la práctica se minimiza $\sum_{i\in\text{zona no tubular}}(y_i - f(\mathbf{x_i})^2$ dada la complejidad en el manejo del valor absoluto.

[2] Observar que la zona tubular juega en el problema de regresión el mismo papel que el margen $\tau = \frac{1}{\|\mathbf{w}\|}$ en el problema de clasificación.

Los ejemplos para los que $\alpha_i{}^+ = \alpha_i{}^- = 0$, no son considerados vectores soporte, se encuentran dentro de la región tubular.

Kernelización de las SVR

Cuando los ejemplos no pueden ajustarse por una función lineal, recurrimos al igual que en el problema de clasificación, al uso de funciones Kernel. A través de un Kernel adecuado se induce un espacio Hilbert, denominado también espacio de características, en este sí es posible ajustar los ejemplos transformados mediante un regresor lineal, que sera de la forma:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\alpha_i{}^- - \alpha_i{}^+) K(\mathbf{x}, \mathbf{x_i}). \tag{2.8}$$

Los coeficientes $\alpha_i{}^+, \alpha_i{}^-$se obtienen como resultado de resolver el problema dual que resulta de (2.6) con los productos escalares sustituidos por las funciones Kernel.

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{n} (\alpha_i{}^- - \alpha_i{}^+) y_i - \epsilon \sum_{i=1}^{n} (\alpha_i{}^- + \alpha_i{}^+) - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i{}^- - \alpha_i{}^+)(\alpha_j{}^- - \alpha_j{}^+) K(\mathbf{x_i}, \mathbf{x_j}) \\ s.a \quad & \sum_{i=1}^{n} (\alpha_i{}^+ - \alpha_i{}^-) = 0, \\ & 0 \leq \alpha_i{}^+, \alpha_i{}^- \leq C, \ i = 1, \ldots, n. \end{aligned} \tag{2.9}$$

En problemas de clasificación tenemos que elegir un Kernel adecuado y un parámetro C. Para resolver problemas de regresión mediante SVRs a lo anterior, se añade la selección de un adecuado ϵ. Ambos parámetros C y ϵ afectan a la complejidad del modelo. En el caso de problemas de regresión con ruido, el parámetro ϵ debería ser elegido de forma que refleje la varianza del ruido de los datos. Puesto que en la mayoría de casos prácticos es posible obtener una medida aproximada de la varianza del ruido a partir de los datos de entrenamiento. Para problemas de regresión sin ruido (problemas de interpolación) el valor ϵ corresponde a la exactitud preestablecida de interpolación, de forma que, cuanto mayor sea el valor de ϵ, menor número de vectores soporte se necesitarán, y viceversa. Por otro lado, la metodología usada para seleccionar los valores óptimos de C y el resto de parámetros del kernel, se basa normalmente en técnicas de validación cruzada al igual que en clasificación.

Capítulo 3

Aplicaciones prácticas

3.1. Introducción al software para SVM

Los clasificadores SVM son uno de los métodos mas populares y eficientes de clasificación y regresión actualmente disponibles, su implementación existe en prácticamente todos los lenguajes de programación mas usados.

Las variaciones de SVM para los que existe solución implementada son los siguientes:

- C-clasificación: la descrita en el capítulo 1 con el parámetro C que refleja el compromiso entre el error de entrenamiento y la complejidad.
- ν-clasificación, este modelo permite un mayor control sobre el numero de vectores soporte especificando un parámetro adicional ν que aproxima la proporción de vectores soporte.
- Clasificación de clase única (novelty detection). Este modelo trata de encontrar el soporte de una distribución y por lo tanto permite la detección de valores atípicos.
- clasificación multiclase, vista en el capítulo 2.
- ϵ-regresión, es la presentada en el capítulo 2.
- ν-regresión, con modificaciones en el modelo, análogas al caso de ν- clasificación.

En cuanto a los kernels, el gaussiano y el laplaciano son generalmente usados cuando no hay ningún conocimiento previo del conjunto de datos dada su validez general. El kernel lineal es útil cuando tratamos grandes vectores con datos muy dispersos, como es el problema de la categorización de un texto. El kernel polinomial es comúnmente usado para el procesamiento de imagenes y el sigmoide para redes neuronales. Por último los splines y el kernel anova se suelen emplear en problemas de regresión. (ver [KMH2006])

La mayor parte del software existente esta escrito en C o C++, como la biblioteca **libsvm** [M2015], ganadora del premio 'IJCNN 2001 Challenge', que proporciona una robusta y rápida implementación de SVM y produce resultados muy competentes en la mayoría de problemas de regresión y clasificación. También destacan **SVMlight**, **SVMTorch**, **Royal Holloway Support Vector Machines**, **mySVM** y **M-SVM**. Otras bibliotecas proveen de interfaces para MATLAB como **The MathWorks**, **SVM and Kernel Methods Matlab Toolbox** o **MATLAB Support Vector Machine Toolbox** y la **SVM toolbox for Matlab** (para más detalle ver [KMH2006]).

Actualmente existen en **R** cuatro bibliotecas con las funciones necesarias para estimar SVM.

- La primera implementación de SVM en **R** se introdujo en la biblioteca **e1071** que proporciona una interfaz para libsvm. Incluye algoritmos para la resolución del problema C-SVM clasificación, ν-clasificación. En regresión para ϵ- clasificación, ν-clasificación y novelty detection. Para problemas multiclase tenemos los algoritmos de 1VR y 1V1. Estos problemas los resuelve a través del optimizador SMO (consultar Apéndice A).

- La biblioteca **kernlab** proporciona una interfaz para ksvm. Dispone de una gran cantidad de funciones kernel y permite la resolución de los problemas del apartado anterior y además C-BSVM (C-clasificación con una restricción sobre las cotas) y su análogo en regresión ϵ-BSVM. Además el optimizador utilizado también es SMO.

- **klaR** incluye una interfaz para SVMlight. Esta biblioteca es aplicable a C-SVM para clasificación y ϵ-SVM en regresión. En clasificación multiclase SVMlight usa el método 1VR. Los problemas de optimización cuadrática anteriores se resuelven con el método chuncking (consultar Anexo A).

- Finalmente **svmpath** implementa un algoritmo con un coste computacional mínimo, que resuelve el problema de C-clasificación en el caso binario para todos los valores del parámetro regulador del coste $\lambda = \frac{1}{C}$. En este caso los únicos kernels utilizados son el gaussiano y el polinomial.

Resumiendo **kernlab** es una implementación muy flexible que incluye la gran mayoría de formulaciones y kernels de SVM, sin embargo carece de la herramienta necesaria para la selección de un modelo adecuado. **e1071** incluye una herramienta de selección de modelos, pero no ofrece tanta flexibilidad en la elección del kernel. La biblioteca **klaR** proporciona una interfaz muy básica para SVMlight y esta permitida solo para uso no comercial. **svmpath** se usa fundamentalmente como herramienta exploratoria para encontrar el valor óptimo del parámetro regulador del coste $\lambda = \frac{1}{C}$

3.2. Aplicación al cáncer de mama: datos WDBC

Veamos una aplicación práctica de las máquinas SVM en el campo de la medicina. Disponemos del conjunto de datos wdbc.data que se puede encontrar en el repositorio UCI [UCIMLR]. Estos datos proceden de un estudio realizado en la Universidad de Wisconsin, en 1995 con 569 pacientes diagnosticadas con cáncer de mama. Para cada paciente tenemos 32 variables: el número de historia clínica, el diagnóstico y atributos de interés médico relativos a las células tumorales. Todas las variables son de tipo numérico excepto, la variable dicotómica Diagnosis (V2), que indica el diagnóstico del tumor, maligno M o benigno B. La proporción real de tumores es 357 benignos y 212 malignos (para más detalle consultar Apéndice B).

Nuestro objetivo es a partir de este conjunto de datos construir una máquina SVM que diagnostique con la mayor certeza posible el tipo de tumor que padece una nueva paciente. Para ello usaremos la biblioteca **e1071** disponible en **R**. Comenzaremos realizando una partición del conjunto de datos inicial (dataset), en dos subconjuntos:

- trainset: son los ejemplos con los que entrenamos, es decir, con los que construiremos el modelo. El tamaño de trainset es entorno al 70 % del volumen de ejemplos de dataset y son seleccionados al azar.

- testset: ejemplos con los que comprobaremos la calidad de la máquina obtenida. Es el 30 % restante de dataset.

Las ordenes que usamos en **R** son las siguientes (fijando previamente una semilla para garantizar la reproducibilidad de los resultados obtenidos):

```
index <- 1:nrow(dataset)
set.seed(150516)
testindex <- sample(index, trunc(length(index)*30/100))
testset <- dataset[testindex,]
trainset <- dataset[-testindex,]
```

La búsqueda de los parámetros óptimos, tanto del kernel como el del coste C, se realiza mediante técnicas de validación cruzada con $k = 10$ bloques. La función `tune.svm` crea un mallado de tantas dimensiones como parámetros a optimizar. Para cada tupla del mallado, se divide el conjunto de datos trainset en k bloques. Se ajustan k modelos de SVM considerando como conjuntos de de entrenamiento k-1 bloques, y con el k-ésimo bloque, se prueba el modelo hallado, obteniéndose el error. El error de la validación cruzada para esa tupla de parámetros del mallado, se halla realizando la media aritmética de los errores de los k modelos. La salida que da **R** es la tupla de parámetros del kernel y el coste, que proporcionan el mínimo error.

En este caso, estudiaremos la calidad del clasificador sobre tres tipos de kernels: polinómico, radial y sigmoide. Aplicamos `tune.svm` sobre trainset para obtener los parámetros óptimos propios de cada kernel (las ordenes que mostraremos a lo largo del capítulo son para un kernel radial. El script completo con las sentencias para el kernel sigmoide y polinómico se pueden consultar en el Apéndice C).

```
set.seed(150516)
tuned1 <- tune.svm(V2~., data = trainset, gamma = 10^(-6:2), cost =10^(0:3),
       kernel="radial")
```

Obtenemos la siguiente tabla:

Kernel	Error óptimo	γ	C	τ	p
Radial	0,0225	0,001	100	–	–
Sigmoide	0,025	0,001	1000	-1	–
Polinómico	0,0225	0,01	10	1	2

Cuadro 3.1: parámetros óptimos para cada tipo de kernel seleccionado. **R** denota a los parámetros de la siguiente forma $\gamma \equiv$ gamma, $C \equiv$ cost, $\tau \equiv$ coef0, $p \equiv$ degree

.

Seguidamente usamos la función `svm` para ajustar el modelo óptimo para cada kernel.

```
model1  <- svm(V2~., data = trainset, kernel = "radial", gamma =  0.001,
        cost = 100, probability=TRUE)
```

Para comparar la calidad de cada modelo generado, libre del peligro de un sobreajuste, usamos las funciones `predict` y `tab` aplicadas sobre testset que nos proporcionan una **matriz de confusión**, una herramienta muy útil en la visualización del rendimiento de un algoritmo de aprendizaje estadístico.

```
prediction1 <- predict(model1, testset[,-2],probability = TRUE)
tab1 <- table(pred = prediction1, true = testset[,2])
```

Sin pérdida de generalidad, en el ámbito médico podemos considerar cuatro posibles conceptos asociados a cada elemento de la matriz de confusión:

- **Verdadero Positivo (TP)**, el paciente da positivo en la prueba diagnóstica de la enfermedad, siendo cierto que la padece. En nuestro caso, se le predice tumor maligno, cuando realmente lo tiene.

- **Falso Positivo (FP)**, el paciente da positivo, siendo falso que padece la enfermedad. Se le predice que tiene tumor maligno, cuando tiene tumor benigno.
- **Verdadero negativo (TN)**, el paciente da negativo, siendo verdad que no posee la enfermedad. Se le predice tumor benigno, siendo cierto.
- **Falso negativo (FN)**, el paciente da negativo en la prueba, cuando realmente esta enfermo. Se le predice que tiene tumor maligno, cuando tiene tumor benigno.

Esta información la recoge la matriz de confusión de la siguiente forma:

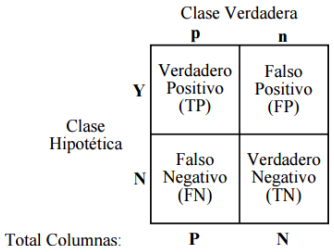

Figura 3.1: matriz de confusión

Ahora, a partir de la matriz mostrada en la Figura 3.1, es posible obtener una serie de métricas útiles para estimar el rendimiento del clasificador:

- **Sensibilidad**: es la probabilidad de que para un sujeto enfermo se obtenga en una prueba diagnóstica un resultado positivo. La sensibilidad caracteriza la capacidad de la prueba para detectar la enfermedad en sujetos enfermos.

$$\text{Sensibilidad} = \frac{TP}{TP+FN}$$

- **Especificidad**: es la probabilidad de que un sujeto sano tenga un resultado negativo en la prueba. La especificidad caracteriza la capacidad de la prueba para detectar la ausencia de la enfermedad en sujetos sanos.

$$\text{Especificidad} = \frac{TN}{TN+FP}$$

- **Precisión**:

$$\text{Precisión} = \frac{TP}{TP+FP}$$

En el cuadro 3.2 podemos comparar nuestros clasificadores en función de la sensibilidad, especificidad y precisión.

Modelo	Sensibilidad	Especificidad	Precisión	AUC
Kernel Radial	0,9531	0,9811	0,9682	0,9847
Kernel Sigmoide	0,9531	0,9716	0,9531	0,9828
Kernel Polinómico	0,9687	0,9716	0,9538	0,9844

Cuadro 3.2: sensibilidad, especificidad, precisión y AUC para cada modelo estimado.

Puesto que los valores anteriores son mejores cuanto mas próximos están a 1, observamos que los tres clasificadores son buenos, sin embargo, para compararlos y decidir cual de ellos es el mejor, necesitamos introducir una nueva herramienta, las **curvas ROC**.

Una curva ROC (Receiver Operating Characteristics), es una descripcion bidimensional del rendimiento de un clasificador, basada en la evolución conjunta de la sensibilidad y la especificidad. El análisis de las curvas ROC ha sido principalmente usado en el área de la medicina con el fin de analizar la utilidad de una prueba diagnóstica. El objetivo es determinar el "punto de corte" para el que se alcanza la sensibilidad y especificidad más alta. Para formular el diagnóstico de una determinada enfermedad, la prueba diagnostica se apoya sobre ese punto de corte a partir del cuál se decide la presencia del diagnostico [F2005]. Recientemente, el uso de curvas ROC se ha generalizado al área del aprendizaje estadístico, debido a que a partir del análisis de las métricas anteriores se obtiene la evaluación del rendimiento de un modelo de clasificación.

Para construir la curva ROC usamos la función `roc` de la biblioteca **pROC**, previamente habremos tomado de `prediction` el bloque de probabilidades relativo a la probabilidad estimada de que la variable diagnosis tome valor M. La razón de tomar probabilidades es que de esta forma podemos hacer un barrido con todos los posibles puntos de corte. Cada punto de corte se corresponde con un punto en la curva ROC, para el que se calcula el **índice de Youden**, definido como Sensibilidad + Especificidad-1. Se elige como punto de corte óptimo el que maximiza el índice anterior. Observar que en la matriz de confusión el punto de corte esta preestablecido en 0.5 de forma que si un ejemplo supera dicho umbral se considera automáticamente que da positivo en la prueba.

```
pred.prob1<-attr(prediction1,"probabilities")[,1]
library(pROC)
rocCurve1 <- roc(response=testset[,2],predictor=pred.prob1)
```

El indicador mas utilizado para la comparación de distintas curvas ROC es el área bajo la curva ROC o **AUC**. El AUC de un clasificador es equivalente a la probabilidad de que el clasificador ordene o puntúe un ejemplo positivo elegido aleatoriamente más alto que uno negativo. La función `auc` nos da el área bajo la curva y `ci.auc` el intérvalo de confianza de este indicador.

```
auc(rocCurve1)
ci.roc(rocCurve1)
```

En medicina un test se considera excelente cuando el AUC esta entre 0.97 y 1. Observando la columna AUC del cuadro 3.2 podemos deducir que la calidad de nuestros clasificadores es excelente, siendo el mejor el de kernel radial.

Por último representamos la curva ROC usando la orden `plot`:

```
plot(rocCurve1,legacy.axes=TRUE, print.thres=TRUE)
```

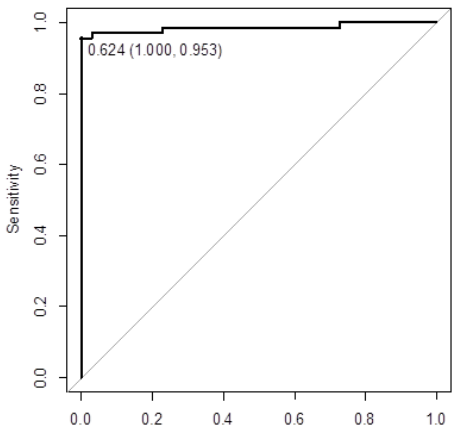

Figura 3.2: Curva ROC obtenida en **R**, representa el rendimiento del modelo con kernel radial.

En la figura 3.2 observamos que el punto de corte para el que se maximiza el índice de Youden es 0.624, para el que se alcanza una sensibilidad y especificidad de 0.953 y 1. Por tanto se produce una ligera mejora respecto a los resultados obtenidos por defecto con el umbral 0.5. En principio podríamos tomar como punto de corte para la prueba diagnóstica el obtenido mediante curvas ROC.

Nuestros resultados, tras haber optimizado por validación cruzada los parámetros del modelo nos dan un rendimiento muy similar al obtenido en [AAS2012], donde se usan métodos de clasificación combinados con el objetivo de mejorar el rendimiento individual de cada uno.

Anexos

Apéndice A

Optimización cuadrática

En este anexo se hace un resumen de las principales ideas de la teoría de optimización, orientado a la resolución de problemas asociados con el uso de SVMs (para mayor detalle consultar [F1987]).

Sean f y g_i, $i = 0,\ldots,n$ funciones convexas, consideramos el siguiente problema de optimización denominado **problema primal**:

$$\begin{array}{lll} \text{mín} & f(\mathbf{x}) & \mathbf{x} \in \Omega \\ s.a & g_i(\mathbf{x}) \leq 0, & i = 1,\ldots,n. \end{array} \tag{A.1}$$

donde f es la función a optimizar y las g_i son las restricciones. La solución del problema primal, $\mathbf{x}^*$, cumplirá que $g_i(\mathbf{x}^*) \leq 0$ y $f(\mathbf{x}^*) \leq f(\mathbf{x})$ $\forall \mathbf{x}$ t.q. $g_i(\mathbf{x}) \leq 0$, donde $i = 1,\ldots,n$. Se define la función de Lagrange como:

$$L(\mathbf{x},\alpha) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{x}),$$

donde los coeficientes $\alpha_i \geq 0$ reciben el nombre de multiplicadores de Lagrange. Esta función incorpora la función objetivo o función a optimizar y las funciones restricción en una única función. A partir de la función de Lagrange se puede definir el **problema dual** como:

$$\begin{array}{lll} \text{máx} & \varphi(\alpha) = \inf\limits_{\mathbf{x}\in\Omega} L(\mathbf{x},\alpha) & \\ s.a & \alpha_i(\mathbf{x}) \geq 0, & i = 1,\ldots,n. \end{array} \tag{A.2}$$

La teoría de la optimalidad asegura que bajo ciertas condiciones resolver el problema dual es equivalente a hallar la solución del primal asociado y viceversa. La ventaja de esta transformación es que normalmente el problema dual es más fácil de resolver que el primal. Los dos siguiente teoremas muestran la relación existente entre las soluciones de los dos problemas.

Teorema. *Sean* $\mathbf{x}$ *y* α *vectores tales que satisfacen las restricciones respectivas del problema primal y dual, es decir,* $g_i(\mathbf{x}) \leq 0$ *y* $\alpha_i \geq 0$*, con* $i = 1,\ldots,n$*, entonces* $\varphi(\alpha) \leq f(\mathbf{x})$.

Del teorema anterior se pueden extraer dos corolarios. El primero establece que el problema dual está acotado superiormente por el problema primal. El segundo permite afirmar que si $\varphi(\alpha) = f(\mathbf{x})$, entonces α y $\mathbf{x}$ son soluciones, respectivamente, del problema dual y primal.

Este teorema establece una heurística para resolver, simultáneamente, el problema primal y dual. De forma que estaremos más cerca de la solución, a medida que la diferencia sea más pequeña. La solución se alcanza cuando la diferencia $|\varphi(\alpha) - f(\mathbf{x})|$ sea más pequeña. La solución se alcanza cuando la diferencia es nula. Esta solución corresponde a un punto silla de la función lagrangiana, caracterizado

por ser simultáneamente un mínimo de $L(\mathbf{x},\alpha)$ respecto de $\mathbf{x}$ y un máximo de $L(\mathbf{x},\alpha)$ respecto de α.

El segundo teorema, denominado teorema de Karush-Kuhn-Tucker establece las condiciones suficientes (condiciones KKT) para que un punto $\mathbf{x}^*$ sea solución del problema primal.

Teorema (de Karush-Kuhn-Tucker). *Si en el problema primal (A.1), las funciones $f:\mathbb{R}^d \longrightarrow \mathbb{R}$ y $g_i:\mathbb{R}^d \longrightarrow \mathbb{R}$, $i=1,\ldots,n$ son todas ellas funciones convexas, y existen constantes $\alpha_i \geq 0$, $i=0,\ldots,n$ tales que:*

$$\begin{aligned} \frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x_j}} + \sum_{i=1}^{n} \alpha_i \frac{\partial g_i(\mathbf{x}^*)}{\partial \mathbf{x_j}} = 0, \qquad & j=1,\ldots,d, \\ \alpha_i^* g_i(\mathbf{x}^*) = 0, \qquad & i=1,\ldots,n, \end{aligned}$$

entonces $(\mathbf{x}^*,\alpha^*)$ es la solución óptima del problema.

La primera condición surge como consecuencia de la definición de la función $\varphi(\alpha)$ como el ínfimo de la función Lagrangiana, punto en el que las derivadas parciales respecto de x deben ser cero. La segunda condición, denominada **condición complementaria**, es la que garantizará que los óptimos del problema primal y dual coincidan $(\varphi(\alpha^*)=f(\mathbf{x}^*))$, ya que, de ser cierta la condición, todos los sumandos del sumatorio de la función Lagrangiana serían nulos.

El interés del Teorema de Karush-Kuhn-Tucker es que establece las condiciones que han de cumplirse para poder resolver el problema primal gracias al dual. Así, partiendo del problema primal, se construye la función lagrangiana. Seguidamente, se aplica la primera condición del teorema de KKT a dicha función y esto permite obtener un conjunto de relaciones que, sustituidas en la función de Lagrange, harán desaparecer todas las variables primales de dicha función. Este paso es equivalente a calcular $\varphi(\alpha)=\text{ínf}_{x\in\Omega} L(\mathbf{x},\alpha)$. La función dual así obtenida, sólo dependerá de los multiplicadores de Lagrange. También es posible que, del conjunto de relaciones obtenido al aplicar la primera condición KKT, surjan restricciones adicionales para las variables duales (multiplicadores de lagrange).

Finalmente la solución del problema primal la obtenemos sustituyendo el resultado obtenido al resolver el problema dual, en las relaciones que anteriormente se obtuvieron al aplicar la primera condición KKT a la función lagrangiana.

En el caso de que la función f sea cuadrática y las g_i restricciones lineales, el problema (A.1) se dice que es de optimización cuadrática:

$$\begin{aligned} \text{mín} \qquad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ s.a \qquad & g_i(\mathbf{x}) = \mathbf{a_i}^T\mathbf{x} + b_i \leq 0, \qquad i=1,\ldots,k, \\ & h_i(\mathbf{x}) = \mathbf{d_i}^T\mathbf{x} + e_i = 0, \qquad i=1,\ldots,l, \end{aligned} \tag{A.3}$$

donde los vectores $\mathbf{x}$, $\mathbf{c}$, $\mathbf{a_i}$ y $\mathbf{d_i} \in \mathbb{R}^d$, la matriz $Q \in \mathbb{R}^{m\times m}$ es semidefinida positiva y b_i y e_i son constantes. Puesto que hay dos tipos de restricciones tenemos dos familias de multiplicadores de Lagrange α_i y β_i, de forma que la función de Lagrange queda:

$$L(\mathbf{x},\alpha,\beta) = f(\mathbf{x}) + \sum_{i=1}^{k} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{l} \beta_i h_i(\mathbf{x}),$$

donde $\alpha=(\alpha_1,\ldots,\alpha_k)^T$, $\alpha_i \geq 0$ para $i=1,\ldots,k$ y $\beta=(\beta_1,\ldots,\beta_l)^T$.

Cuando el problema de optimización es cuadrático, como caso particular del Teorema de Karush-Kuhn-Tucker se deduce el siguiente corolario:

Corolario. *Sea el problema de optimización cuadrática (A.3), la solución óptima* $(\mathbf{x}^*, \alpha^*, \beta^*)$ *existe si y solo si se satisfacen las siguientes condiciones:*

$$\begin{aligned}
\frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x_j}} + \sum_{i=1}^{n} \alpha_i \frac{\partial g_i(\mathbf{x}^*)}{\partial \mathbf{x_j}} + + \sum_{i=1}^{n} \beta_i \frac{\partial h_i(\mathbf{x}^*)}{\partial \mathbf{x_j}} = \mathbf{0}, \qquad & j = 1, \ldots, m, \\
\alpha_i^* g_i(\mathbf{x}^*) = 0, \qquad & i = 1, \ldots, k, \\
\alpha_i^* \geq 0, \qquad & i = 1, \ldots, k, \\
h_i(x^*) = 0, \qquad & i = 1 \ldots, l.
\end{aligned}$$

Una vez hemos transformado el problema (A.3) a su dual se procede a su resolución mediante técnicas de programación cuadrática. Realizamos una breve reseña sobre algunos de los algoritmos más utilizados, para mayor detalle consultar [KMH2006],[AFP].

- El método chunking, permite resolver un problema de optimización cuadrático con un pequeño conjunto (chunk) de variables o ejemplos de entrenamiento en nuestro caso. Si particularizamos a la resolución de problemas cuadráticos de SVMs, los ejemplos cuyos multiplicadores de Lagrange asociados, α_i, toman valor 0 o C, son excluidos y el problema de optimización se resuelve con el resto de ejemplos. Este procedimiento se repite hasta que se alcanza la solución óptima del problema cuadrático.

- SMO (Sequential Minimization Optimization). Este algoritmo esta basado en el método anterior con la diferencia de que ahora, en cada iteración, se resuelve de forma analítica un problema cuadrático usando únicamente dos ejemplos (conjunto activo). De esta forma en cada iteración se actualizan los valores de los α_i correspondientes a los ejemplos para los que se ha resuelto el problema. Actualmente, las bibliotecas más potentes, con las funciones necesarias para estimar SVM, utilizan SMO como optimizador cuadrático.

Apéndice B

Descripción variables del conjunto de datos

Los datos de Wisconsin Diagnostic Breast Cancer (WDBC) fueron recogidos para un estudio realizado por el Dr. William H. Wolberg, W. Nick Street y Olvi L. Mangasarian, investigadores de la universidad de Wisconsin en Noviembre de 1995. El objetivo de dicho estudio consistía en diagnosticar mediante técnicas de aprendizaje estadístico el tipo de tumor de mama de cada paciente dependiendo de las características de los núcleos celulares extraídos.

Para cada uno de los 569 pacientes se tomó una imagen de tejido tumoral procedente de la mama. De las 32 variables del conjunto de datos las dos primeras son relativas a la identificación del paciente y el diagnóstico de su tumor:

- Número de historia clínica.
- Diagnóstico o diagnosis del tumor del paciente (M = maligno, B = benigno). Esta variable se denota V2 en el conjunto de datos dataset, indica la etiqueta de cada ejemplo.

En cada imagen se recogió información sobre 10 características de cada núcleo celular del tejido:

- radio, es la media de las distancias del centro a los puntos de su perímetro.
- textura, es la desviación estándar de los valores de la escala de gray.
- perímetro.
- área.
- suavidad, es la variación local de las longitudes del radio.
- compactitud, definida por $\frac{\text{perímetro}^2}{\text{area}-1}$.
- concavidad, es la severidad de las porciones cóncavas del contorno.
- puntos cóncavos, es el número de porciones cóncavas del contorno.
- simetría.
- dimensión fractal, definida por $coastline\ approximation - 1$.

Los datos que visualizamos se obtuvieron computando para cada una de las características anteriores la media, la desviación típica y el peor valor de los núcleos de la imagen, de forma que obtenemos 30 variables. A modo de ejemplo la variable 3 indica la media de los radios, la 13 la desviación estandar de los radios y la 23 el peor de los radios en la imagen de cada paciente. Además cada uno de los valores que toman las variables esta codificado con 4 dígitos y no hay datos ausentes.

Apéndice C

Script R y salidas no incluidas en la memoria

C.1. Script

```
dataset <- read.csv("C:/Users/Vaio/Desktop/TFG/aplicaciones practicas
  TFG/dataset.txt", header=FALSE)
View(dataset)

#separamos dataset en un conjunto de entrenamiento y otro de prueba, fijando
una semilla

index <- 1:nrow(dataset)
set.seed(150516)
testindex <- sample(index, trunc(length(index)*30/100))

testset <- dataset[testindex,]

trainset <- dataset[-testindex,]

#instalamos el paquete e1071 y lo cargamos

install.packages('e1071', dependencies = TRUE)
library(e1071)

#para el kernel radial hallamos los parámetros gamma y coste usando la técnica
de validación cruzada
#fijamos la semilla antes de cada simulación

set.seed(150516)

tuned1 <- tune.svm(V2~., data = trainset, gamma = 10^(-6:2), cost = 10^(-1:3),
      kernel="radial")
summary(tuned1)

#obtenemos para el kernel radial los parámetros optimos gamma=0.001,coste=100
#para ellos, creamos el modelo tomando probability=TRUE así podremos hacer
un barrido con todos los puntos de corte en las curvas ROC.
```

```
model1  <- svm(V2~., data = trainset, kernel = "radial", gamma =  0.001 ,
        cost = 100, probability=TRUE)

summary(model1)

#usando la orden predict y tab construimos la matriz de confusión

prediction1 <- predict(model1, testset[,-2],probability = TRUE)
tab1 <- table(pred = prediction1, true = testset[,2])
show(tab1)

#cogemos la parte de probabilidades de prediction y la primera columna
asociada a M

pred.prob1<-attr(prediction1,"probabilities")[,1]

#construimos la curva ROC

library(pROC)
rocCurve1 <- roc(response=testset[,2],predictor=pred.prob1)
auc(rocCurve1)
ci.roc(rocCurve1)
plot(rocCurve1,legacy.axes=TRUE,print.thres=TRUE)

#kernel sigmoide

set.seed(150516)
tuned2 <- tune.svm(V2~., data = trainset, gamma = 10^(-6:2), cost =10^(-1:3),
       coef0=(-2:2),kernel="sigmoid")
summary(tuned2)

model2  <- svm(V2~., data = trainset, kernel = "sigmoid", gamma =  0.001 ,
       cost = 1000, coef0=-1, probability=TRUE)
summary(model2)

prediction2 <- predict(model2, testset[,-2],probability = TRUE)
tab2 <- table(pred = prediction2, true = testset[,2])
show(tab2)

pred.prob2<-attr(prediction2,"probabilities")[,1]

library(pROC)
rocCurve2 <- roc(response=testset[,2],predictor=pred.prob2)
auc(rocCurve2)
ci.roc(rocCurve2)
plot(rocCurve2,legacy.axes=TRUE,print.thres=TRUE)

#kernel polinomial

set.seed(150516)
```

```
tuned3 <- tune.svm(V2~., data = trainset, gamma = 10^(-5:0), cost =10^(1:3),
      coef0=(-1:1),degree=(0:4),kernel="polynomial")
summary(tuned3)

model3  <- svm(V2~., data = trainset, kernel = "polynomial", gamma =  0.01 ,
       cost = 10, coef0=1, degree=2 , probability=TRUE)
summary(model3)

prediction3<- predict(model3, testset[,-2],probability = TRUE)
tab3 <- table(pred = prediction3, true = testset[,2])
show(tab3)

pred.prob3<-attr(prediction3,"probabilities")[,1]

library(pROC)
rocCurve3 <- roc(response=testset[,2],predictor=pred.prob3)
auc(rocCurve3)
ci.roc(rocCurve3)
plot(rocCurve3,legacy.axes=TRUE,print.thres=TRUE)
```

C.2. Salidas no incluidas en la memoria

```
Radial

> library(e1071)
> set.seed(150516)
> tuned1 <- tune.svm(V2~., data = trainset, gamma = 10^(-6:2),
        cost = 10^(-1:3),kernel="radial")
> summary(tuned1)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.001  100

- best performance: 0.0225

- Detailed performance results:
   gamma  cost     error dispersion
1  1e-06 1e-01 0.3709615 0.07259428
2  1e-05 1e-01 0.3709615 0.07259428
3  1e-04 1e-01 0.3709615 0.07259428
4  1e-03 1e-01 0.2832051 0.07817465
5  1e-02 1e-01 0.0525000 0.03809710
6  1e-01 1e-01 0.0775641 0.04624279
7  1e+00 1e-01 0.3709615 0.07259428
8  1e+01 1e-01 0.3709615 0.07259428
9  1e+02 1e-01 0.3709615 0.07259428
```

```
10 1e-06 1e+00 0.3709615 0.07259428
11 1e-05 1e+00 0.3709615 0.07259428
12 1e-04 1e+00 0.2755769 0.08212662
.
.
.

> model1  <- svm(V2~., data = trainset, kernel = "radial",
         gamma =  0.001 , cost = 100, probability=TRUE)
> summary(model1)

Call:
svm(formula = V2 ~ ., data = trainset, kernel = "radial", gamma = 0.001,
    cost = 100, probability = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  0.001

Number of Support Vectors:  39

 ( 19 20 )


Number of Classes:  2

Levels:
 B M
> prediction1 <- predict(model1, testset[,-2],probability = TRUE)
> tab1 <- table(pred = prediction1, true = testset[,2])
> show(tab1)
    true
pred   B   M
   B 104   3
   M   2  61

#Sensibilidad=61/(61+3)= 0,9531
#Especificidad=104/(104+2)= 0,9811
#Precisión=61/(61+2)= 0,9682

> pred.prob1<-attr(prediction1,"probabilities")[,1]
> library(pROC)
> rocCurve1 <- roc(response=testset[,2],predictor=pred.prob1)
> auc(rocCurve1)
Area under the curve: 0.9847
> ci.roc(rocCurve1)
95\% CI: 0.9614-1 (DeLong)
>  plot(rocCurve1,legacy.axes=TRUE,print.thres=TRUE)
```

```
Call:
roc.default(response = testset[, 2], predictor = pred.prob1)

Data: pred.prob1 in 106 controls (testset[, 2] B) < 64 cases (testset[, 2] M).
Area under the curve: 0.9847
```

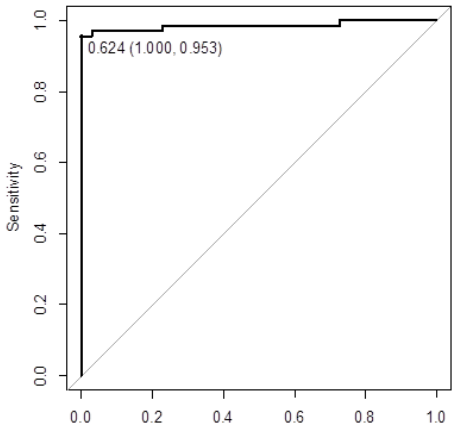

Figura C.1: Curva ROC obtenida en **R**, representa de rendimiento del modelo con kernel radial.

```
Sigmoide

> set.seed(150516)
> tuned2 <- tune.svm(V2~., data = trainset, gamma = 10^(-6:2), cost =10^(-1:3),
         coef0=(-2:2),kernel="sigmoid")
> summary(tuned2)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma coef0 cost
 0.001    -1 1000

- best performance: 0.025

- Detailed performance results:
    gamma coef0  cost      error dispersion
1   1e-06    -2 1e-01 0.37096154 0.07259428
2   1e-05    -2 1e-01 0.37096154 0.07259428
3   1e-04    -2 1e-01 0.37096154 0.07259428
4   1e-03    -2 1e-01 0.37096154 0.07259428
5   1e-02    -2 1e-01 0.36339744 0.07649301
6   1e-01    -2 1e-01 0.10012821 0.04393591
7   1e+00    -2 1e-01 0.09769231 0.04770196
8   1e+01    -2 1e-01 0.08269231 0.02893860
9   1e+02    -2 1e-01 0.08256410 0.04081006
10  1e-06    -1 1e-01 0.37096154 0.07259428
```

```
11  1e-05    -1 1e-01 0.37096154 0.07259428
12  1e-04    -1 1e-01 0.37096154 0.07259428
.
.
.

> model2  <- svm(V2~., data = trainset, kernel = "sigmoid", gamma =  0.001,
          cost = 1000, coef0=-1, probability=TRUE)
> summary(model2)

Call:
svm(formula = V2 ~ ., data = trainset, kernel = "sigmoid", gamma = 0.001,
    cost = 1000, coef0 = -1, probability = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  sigmoid
       cost:  1000
      gamma:  0.001
     coef.0:  -1

Number of Support Vectors:  34

 ( 15 19 )


Number of Classes:  2

Levels:
 B M
> prediction2 <- predict(model2, testset[,-2],probability = TRUE)
> tab2 <- table(pred = prediction2, true = testset[,2])
> show(tab2)
    true
pred   B   M
   B 103   3
   M   3  61

#Sensibilidad=61/(61+3)= 0,953125
#Especificidad =103/(103+3)= 0,9716
#Precisión=61/(61+3)= 0,953125

> pred.prob2<-attr(prediction2,"probabilities")[,1]

> library(pROC)
> rocCurve2 <- roc(response=testset[,2],predictor=pred.prob2)
> auc(rocCurve2)
Area under the curve: 0.9828
> ci.roc(rocCurve2)
95\% CI: 0.9581-1 (DeLong)
```

```
> plot(rocCurve2,legacy.axes=TRUE,print.thres=TRUE)

Call:
roc.default(response = testset[, 2], predictor = pred.prob2)

Data: pred.prob2 in 106 controls (testset[, 2] B) < 64 cases (testset[, 2] M).
Area under the curve: 0.9828
```

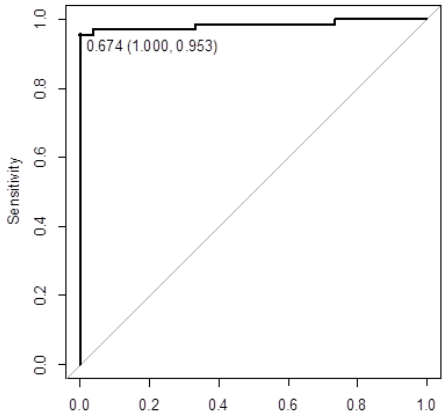

Figura C.2: Curva ROC obtenida en **R**, representa de rendimiento del modelo con kernel sigmoide.

```
Polinomial

tuned3 <- tune.svm(V2~., data = trainset, gamma = 10^(-5:0), cost =10^(1:3),
coef0=(-1:1),degree=(0:4),kernel="polynomial")
> summary(tuned3)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 degree gamma coef0 cost
      2  0.01     1   10

- best performance: 0.0225

- Detailed performance results:
    degree gamma coef0 cost      error dispersion
1        0 1e-05    -1   10 0.37096154 0.07259428
2        1 1e-05    -1   10 0.36596154 0.07675757
3        2 1e-05    -1   10 0.37096154 0.07259428
4        3 1e-05    -1   10 0.15269231 0.06584304
5        4 1e-05    -1   10 0.38852564 0.05716796
6        0 1e-04    -1   10 0.37096154 0.07259428
7        1 1e-04    -1   10 0.06250000 0.04750731
8        2 1e-04    -1   10 0.65685897 0.07181271
9        3 1e-04    -1   10 0.05000000 0.03535534
10       4 1e-04    -1   10 0.69942308 0.05443097
```

```
11       0 1e-03    -1   10 0.37096154 0.07259428
12       1 1e-03    -1   10 0.03000000 0.02581989
.
.
.

> model3  <- svm(V2~., data = trainset, kernel = "polynomial", gamma =  0.01,
         cost = 10, coef0=1, degree=2 , probability=TRUE)
> summary(model3)

Call:
svm(formula = V2 ~ ., data = trainset, kernel = "polynomial", gamma = 0.01,
    cost = 10, coef0 = 1, degree = 2, probability = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  10
     degree:  2
      gamma:  0.01
     coef.0:  1

Number of Support Vectors:  43

 ( 18 25 )


Number of Classes:  2

Levels:
 B M
> prediction3<- predict(model3, testset[,-2],probability = TRUE)
> tab3 <- table(pred = prediction3, true = testset[,2])
> show(tab3)
    true
pred   B   M
   B 103   2
   M   3  62
#Sensibilidad =62/(62+2)=0.9687
#Especificidad =103/(103+3)=0.9716
#Precisión=62/(62+3)=0.9538
> pred.prob3<-attr(prediction3,"probabilities")[,1]
> library(pROC)
> rocCurve3 <- roc(response=testset[,2],predictor=pred.prob3)
> auc(rocCurve3)
Area under the curve: 0.9844
> ci.roc(rocCurve3)
95\% CI: 0.9609-1 (DeLong)
> plot(rocCurve3,legacy.axes=TRUE,print.thres=TRUE)
```

```
Call:
roc.default(response = testset[, 2], predictor = pred.prob3)

Data: pred.prob3 in 106 controls (testset[, 2] B) < 64 cases (testset[, 2] M).
Area under the curve: 0.9844
```

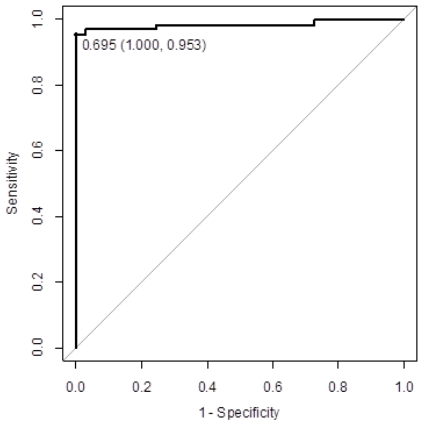

Figura C.3: Curva ROC obtenida en **R**, representa de rendimiento del modelo con kernel polinomial.

Bibliografía

[AAS2012] M.B.ABDELHALIM, M.ABD-ELGHANY ZEID, G.I. SALAMA, *Breast Cancer Diagnosis on Three Different Datasets Using Multi-Classifiers*,International Journal of Computer and Information Technology, 1, 1, September 2012.

[A1944] N. ARONSZAJN, *La théorie générale des noyaux réproduisants et ses applications.* Proceedings of the Cambridge Philosophical Society, 39, 133–153, 1944.

[AFP] A.ARTÉS,A.R.FIGUEIRAS Y F.PÉREZ, *Double Chunking for Solving SVMs for Very Large Datasets*

[BD1995] G. BAKIRI Y T. DIETTERICH *Solving multiclass learning problems via error-correcting output codes.* J. Artif. Intell. Res. 2, 263–286, 1995.

[BGV1992] B.E. BOSER, I.M. GUYON, AND V.N. VAPNIK, *A training algorithm for optimal margin classifiers* Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, 144–152, 1992.

[BL] L. BOTTOU Y CH. LIN, *Support Vector Machine Solvers*

[C2014] E. CARMONA, *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia, Madrid, 2014.

[CV1995] C. CORTES AND V.N. VAPNIK. *Support vector networks.* Machine Learning, 20, 1–25, 1995.

[VCh1989] A.JA. CHERVONENKIS Y V.N. VAPNIK *The necessary and sufficient conditions for consistency of the method of empirical risk minimization.* Yearbook of the Academy of Sciences of the USSR on recognition, Classification, and Forecasting 2, Moscow 207-249, 1989.

[CS2001] K. CRAMMER, Y. SINGER *On the algorithmic implementation of multiclass kernel-based vector machines.* J. Mach. Learn. 2, 265–292, 2001.

[F2005] T. FAWCET *An Introduction to ROC Analysis.* In Pattern Recognition Letters, 861-874, 2005.

[F1987] R.FLETCHER *Practical Methods of Optimization.* John Wiley and Sons, Chichester,second edition, 1987.

[KMH2006] K. HORNIK, A. KARATZOGLOU Y D. MEYER, *Support Vector Machines in R*, Journal of Statistical Software, 15, 9, April 2006.

[J1982] K.JÖRGENS *Linear integral operators, Pitman, Boston, 1982*

[VL1963] A. LERNER Y V.N. VAPNIK. *Pattern recognition using generalized portrait method.* Automation and Remote Control, 24, 774–780, 1963.

[M2015] D. MEYER, *Support Vector Machines The Interface to libsvm in package e1071*, FH Technikum Wien, Austria, 2015.

[V1995] V.N. VAPNIK. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[V1998] V.N. VAPNIK *Statistical Learning Theory.* Wiley, New York, 1998.

[WX2014] Z. WANG Y X. XUE, *Support Vector Machines Applications*, Springer, 2014.

[WW1999] C.WATKINS Y J. WESTON, *Multi-class support vector machines* Proceedings of ESANN99, 1999.

[WSVM] MÁQUINAS DE VECTORES DE SOPORTE, `https://es.wikipedia.org/wiki/Máquinas_de_vectores_de_soporte`.

[WAS] APRENDIZAJE SUPERVISADO, `https://es.wikipedia.org/wiki/Aprendizaje_supervisado`

[UCIMLR] UCI, MACHINE LEARNING REPOSITORY, `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`